

TOPOLOGY AND DYNAMICS OF AN ARTIFICIAL  
GENETIC REGULATORY NETWORK MODEL

CENTRE FOR NEWFOUNDLAND STUDIES

---

**TOTAL OF 10 PAGES ONLY  
MAY BE XEROXED**

(Without Author's Permission)

P. DWIGHT KUO











# Topology and Dynamics of an Artificial Genetic Regulatory Network Model

by

© P. Dwight Kuo

A thesis submitted to the  
School of Graduate Studies  
in partial fulfilment of the  
requirements for the degree of  
Master of Science

Department of Computer Science  
Memorial University of Newfoundland

July 2005

St. John's

Newfoundland



## Abstract

This thesis presents some of the methods of studying models of regulatory networks using mathematical and computational formalisms. A basic review of the biology behind gene regulation is introduced along with the formalisms used for modelling networks of such regulatory interactions. Topological measures of large-scale complex networks are discussed and then applied to a specific artificial regulatory network model created through a duplication and divergence mechanism. Such networks share topological features with natural transcriptional regulatory networks. Thus, it may be the case that the topologies inherent in natural networks may be primarily due to their method of creation rather than being exclusively shaped by subsequent evolution under selection.

The evolvability of the dynamics of these networks are also examined by evolving networks in simulation to obtain three simple types of output dynamics. The networks obtained from this process show a wide variety of topologies and numbers of genes indicating that it is relatively easy to evolve these classes of dynamics in this model.



## Acknowledgements

In no particular order, I would like to kindly thank François Képès of Atelier de Génomique Cognitive, CNRS, Nadav Kashtan and Ron Milo of the Weizmann Institute of Science, Mike Hallett, Leon Glass, Theodore Perkins and Peter Swain of McGill University, William Langdon of University College London, André Leier of the University of Queensland, and Todd Wareham of Memorial University of Newfoundland for helpful discussions on the work described in this thesis. I would also like to acknowledge the Atlantic Computational Excellence Network (ACEnet) for their technical support and for providing the computers on which many of the simulations described herein have been performed.

Thanks also to my parents, family and friends for their enthusiastic support, kindness and understanding during these busy years. Most of all, I would like to thank my supervisor, Wolfgang Banzhaf, for his continued guidance and support without which this work would not have been completed.

Portions of this work have been previously published in *Journal of Biological Physics and Chemistry*, *Proceedings of the 9<sup>th</sup> International Conference on the Simulation and Synthesis of Living Systems* and *Lecture Notes in Computer Science*, Vol. 3242 in collaboration with Wolfgang Banzhaf and André Leier. Portions of this thesis have also been presented at the Centre for Nonlinear Dynamics in Physiology and Medicine, McGill University, the 5<sup>th</sup> Numerical Analysis Day, Acadia University and the Dept. of Computer Science Graduate Student Research Forum (2004, 2005), Memorial University of Newfoundland and have also appeared in project reports at Memorial University of Newfoundland in *CS6754: Post-Genomic Computational Biology*, *CMSC6910: Matrix Computations & Applications* and *CMSC6930: Numerical Linear Algebra for High-Performance Computing*.



# Contents

Abstract	ii
Acknowledgements	iii
List of Tables	viii
List of Figures	x
List of Algorithms	xiii
1 Introduction	1
2 Gene Regulation	4
2.1 Basics of Gene Expression . . . . .	5
2.1.1 DNA . . . . .	5
2.1.2 The Central Dogma . . . . .	6
2.1.3 Transcription . . . . .	7
2.1.4 Translation . . . . .	9
2.2 Models and Mechanisms of Gene Regulation . . . . .	10
2.2.1 Operon Model of Gene Regulation . . . . .	11
2.2.2 Control of Gene Expression in Eukaryotes . . . . .	15



2.2.3	RNA interference (RNAi) and microRNA (miRNA) . . . . .	19
2.3	Genetic Perturbation Experiments . . . . .	21
2.4	Caveat Emptor with mRNA Measurements . . . . .	24
2.5	Conclusion . . . . .	26
<b>3</b>	<b>Network Models</b>	<b>27</b>
3.1	Boolean Network Models . . . . .	28
3.2	Differential Equation Models . . . . .	31
3.2.1	Ordinary Differential Equations . . . . .	32
3.2.2	Weight Matrices . . . . .	32
3.2.3	Piece-wise Linear Differential Equation Model . . . . .	34
3.2.4	S-Systems . . . . .	36
3.2.5	Control Theory / State Space Models . . . . .	37
3.3	Stochastic Models . . . . .	38
3.4	Artificial Regulatory Network Model . . . . .	40
3.5	Conclusion . . . . .	43
<b>4</b>	<b>Topological Characterization</b>	<b>44</b>
4.1	Scale-Free Network Topologies . . . . .	45
4.2	Small-World Network Topologies . . . . .	48
4.3	Network Motifs . . . . .	49
4.4	Conclusion . . . . .	51
<b>5</b>	<b>Network Topologies in the ARN Model</b>	<b>52</b>
5.1	Gene Duplication and the ARN Model . . . . .	53
5.2	Scale-Free & Small-World Topologies in the ARN Model . . . . .	55
5.2.1	Results . . . . .	59



5.2.2	Analysis . . . . .	61
5.3	Network Motifs in the ARN Model . . . . .	66
5.3.1	Results . . . . .	66
5.3.2	Analysis . . . . .	69
5.4	Conclusion . . . . .	71
<b>6</b>	<b>Evolving Dynamics in the ARN Model</b>	<b>73</b>
6.1	Extracting a Signal from the ARN Model . . . . .	74
6.2	Evolutionary Strategies . . . . .	76
6.3	Optimization and Simulation Details . . . . .	76
6.4	Results . . . . .	79
6.5	Analytical Considerations . . . . .	80
6.6	Conclusion . . . . .	84
<b>7</b>	<b>Conclusion</b>	<b>85</b>
	<b>References</b>	<b>87</b>
<b>A</b>	<b>Measurement Technologies</b>	<b>103</b>
A.1	Genetic Microarrays . . . . .	103
A.2	Serial Analysis of Gene Expression (SAGE) . . . . .	104
<b>B</b>	<b>Determining Interactions Between Genes</b>	<b>106</b>
B.1	Correlation Analysis . . . . .	106
B.2	Clustering Methods . . . . .	107
B.2.1	$k$ -means Algorithm . . . . .	107
B.2.2	Hierarchical Clustering . . . . .	108
B.2.3	Support Vector Machines . . . . .	108



B.3	Projection Methods . . . . .	109
B.3.1	Principal Component Analysis (PCA) . . . . .	109
B.3.2	Independent Component Analysis (ICA) . . . . .	110
C	Subgraph Finding Algorithm	111
C.1	Algorithm Implementation . . . . .	111
C.2	Algorithm Pseudocode . . . . .	114
D	Parallelizing Motif Search	117
D.1	Parallelization of the Subgraph Algorithm . . . . .	117
D.2	Comparison of Serial and Parallel Implementations of the Algorithm .	121
D.2.1	Shared Memory Algorithm . . . . .	122
D.2.2	Distributed Memory Algorithm . . . . .	124
D.3	Conclusions . . . . .	125
E	Subgraphs of Size Three	127
F	Subgraph Counts for Size Three	129
G	Subgraphs of Size Four	131
H	Subgraphs Counts for Size Four	134
I	Evolving Networks with a Restricted Number of Genes	137



# List of Tables

5.1	Sum of square error (SSE) between the distributions of subgraph counts (for subgraph size three) for the four types of networks examined. . . .	69
5.2	Sum of square error (SSE) between the distributions of subgraph counts (for subgraph size four) for the four types of networks examined. . . .	69
6.1	Results of 10 runs of $(50 + 100)$ -ES on Case #1 (sinusoid). . . . .	80
6.2	Results of 10 runs of $(50 + 100)$ -ES on Case #2 (exponential). . . . .	81
6.3	Results of 10 runs of $(50 + 100)$ -ES on Case #3 (sigmoid). . . . .	82
C.1	A comparison of the different data structure implementations considered.	112
E.1	Network motifs of size three and their ID. . . . .	128
F.1	Subgraphs of size three and their distribution. . . . .	130
G.1	Subgraphs of size four and their ID. . . . .	132
G.2	Subgraphs of size four and their ID. . . . .	133
H.1	Subgraphs of size four and their distribution. . . . .	135
H.2	Subgraphs of size four and their distribution. . . . .	136
I.1	Results of 10 runs of $(50 + 100)$ -ES on Case #1 (sinusoid) with a penalty function. . . . .	138



I.2	Results of 10 runs of (50 + 100)-ES on Case #2 (exponential) with a penalty function. . . . .	138
I.3	Results of 10 runs of (50+100)-ES on Case #3 (sigmoid) with a penalty function. . . . .	138



# List of Figures

2.1	A DNA molecule. . . . .	5
2.2	The central dogma. . . . .	7
2.3	Table of the mapping between mRNA triplets and amino acids. . . .	9
2.4	The lactose operon. . . . .	13
2.5	The lactose operon. . . . .	13
2.6	The tryptophan operon. . . . .	15
2.7	An mRNA created from a single strand of DNA. . . . .	17
3.1	An example of a Boolean network. . . . .	29
3.2	The state-transition diagram for the Boolean network. . . . .	30
3.3	A Boolean network model of the repressilator. . . . .	35
3.4	The relation between the matrices <b>A</b> , <b>B</b> , <b>C</b> , <b>D</b> , the state vector, $\mathbf{x}(t)$ , the output vector, $\mathbf{y}(t)$ and the input vector, $\mathbf{u}(t)$ in control theory. . .	37
3.5	Bit string for one gene in the ARN model. . . . .	42
4.1	Examples of two different degree distributions. . . . .	46
4.2	An example of a network displaying scale-free characteristics. . . . .	46
4.3	Networks generated from a regular lattice (left-most graph). . . . .	48
5.1	Whole genome duplication and divergence. . . . .	53
5.2	Gene-protein interaction networks generated by two different thresholds.	55



5.3	Diagram showing the fraction of edges in a graph at a given threshold ( $x$ -axis) compared to a fully connected graph for 200 networks generated by duplication and divergence. . . . .	56
5.4	Histogram of the number of genes in the ARN model. . . . .	57
5.5	Distribution of values of $\gamma$ . . . . .	60
5.6	Degree distribution of a network generated by duplication and divergence with 1% mutation. . . . .	61
5.7	Plot of $\frac{C}{C_{random}}$ and $\frac{L_{random}-L}{L_{random}}$ for each of the randomly generated genomes (200 genomes) with a mutation rate of 1.0%. . . . .	62
5.8	Plot of $\frac{C}{C_{random}}$ and $\frac{L_{random}-L}{L_{random}}$ for each of the randomly generated genomes (200 genomes) with a mutation rate of 5.0%. . . . .	63
5.9	An example of the effect of two duplication events. . . . .	63
5.10	Demonstrates that any of the nodes in the original topology can be replaced with its copy without changing the topology and vice versa .	65
5.11	Frequency of occurrence for subgraphs of size three. . . . .	67
5.12	Frequency of occurrence for subgraphs of size four. . . . .	68
5.13	The effect of whole genome duplication on the simplest possible interaction between two genes. . . . .	71
6.1	Plot of the three fitness cases. . . . .	78
6.2	The best solution of 10 runs on Case #1. . . . .	79
6.3	The best solution of 10 runs on Case #2. . . . .	80
6.4	The best solution of 10 runs on Case #3. . . . .	81
D.1	Performance of the shared memory algorithm. . . . .	123
D.2	Performance of the distributed memory algorithm. . . . .	125



E.1	“Piled Higher and Deeper” by Jorge Cham, <a href="http://www.phdcomics.com">www.phdcomics.com</a> . Reprinted with permission. . . . .	127
G.1	“Piled Higher and Deeper” by Jorge Cham, <a href="http://www.phdcomics.com">www.phdcomics.com</a> . Reprinted with permission. . . . .	131
I.1	Three two-gene networks that generate sigmoid dynamics. . . . .	139



## List of Algorithms

1	Pseudocode for the evolutionary strategies algorithm. . . . .	77
2	SearchIsomorphism: Find the set of all isomorphisms for all subgraphs of size $n$ . . . . .	114
3	ReduceMatrixtoSmallestInt: Reduction of a matrix into a canonical form integer. . . . .	115
4	ObtainSubgraphCount: Obtain subgraph count for a graph $G$ . . . . .	115
5	DepthFirstSearch: Depth first search algorithm implementation for sub- graph counting. . . . .	116
6	TotalEnumerationSearch: Search the graph, $G$ , using complete brute- force enumeration. . . . .	118
7	Master subroutine for the MPI version of the algorithm. . . . .	120
8	Slave subroutine for the MPI version of the algorithm. . . . .	121



# Chapter 1

## Introduction

Regulatory networks have become an important new area of research in the biological and biomedical sciences (Bower and Bolouri, 2001, Davidson, 2001, Kitano, 2001). With draft sequences of the human genome complete (in addition to other organisms), scientists may now search these sequences for both genes and transcription factor binding sites in order to better understand which genes and proteins interact. It has been recognized that the DNA information controlling gene expression is the key to understanding differences between species and thus to evolution (Hood and Galas, 2003).

Taking these interactions as a whole, a network of interactions (a so-called regulatory network) can be visualized where genes interact by regulating other genes and their products to produce and regulate a myriad of cellular processes and functions. There are three major genetic mechanisms, all tied to regulation (Davidson, 2001) that allow the variety of reactions of living organisms to the pressure for survival: interactions between the products of genes, shifts in the timing of gene expression (heterochrony) and shifts in the location of gene expression (spatial patterning).

These mechanisms allow nature to set up and control the mechanisms of evolution,



development and physiology. Studying models of regulatory networks can help us to understand some of these mechanisms providing lessons for biology and possibly in the area of artificial evolution. This thesis presents some of the methods of studying models of regulatory networks using mathematical and computational formalisms and uses them to pose questions regarding the topological organization of such networks as has been suggested in work such as Kauffman (2004).

This thesis is organized as follows: Chapter 2 introduces the reader to the biological specifics of gene regulation. This includes a brief review of DNA, the processes of transcription and translation as well as the main mechanisms of gene regulation. In addition, some other potentially important mechanisms of gene regulation are reviewed such as the RNA interference effect and miRNAs. The chapter concludes with a description of the role and relevance of studying genetic regulatory networks. This chapter may be skipped by the more biologically-versed reader.

Chapter 3 reviews some of the more common mathematical and computational abstractions for modelling genetic regulatory networks. These include classes of Boolean and differential equation models in addition to the artificial regulatory network (ARN) model of Banzhaf (2003a,b) which is further investigated in this thesis.

Chapter 4 reviews work on the characterization of the topology of complex networks in general with some emphasis on genetic regulatory networks. Specific topics introduced include scale-free and small-world network topologies and network motifs.

Chapter 5 presents work on the topological properties of the ARN model generated by a whole genome duplication and divergence process (which is also introduced). Such networks have scale-free and small-world topologies and have subgraph distributions similar to those of the transcriptional regulatory networks of *Escherichia coli* and *Saccharomyces cerevisiae*.

Chapter 6 presents work on investigating whether the dynamics of the ARN model



can be evolved toward simple dynamics such as that of a sigmoid, sinusoid and exponential decay.

The Appendices present additional data not presented directly in the thesis. Additional work and techniques relevant to the analysis of biological systems are also discussed. Topics covered include techniques for measuring gene expression, methods for elucidating the relationship between genes, algorithm descriptions, and data not presented in the main body of the text.



# Chapter 2

## Gene Regulation

In this chapter, much of the basic biology required to understand gene regulation is introduced. An overview of the process of transcription of DNA to RNA and subsequent translation of RNA to proteins is provided. However, this is presented in the most general terms in large part without regard to whether it occurs in a prokaryote<sup>1</sup> or eukaryote<sup>2</sup>. Of course, there are major differences in how this process is carried out in both types of organism. Many of these differences will be discussed further in the chapter in relation to how regulation exploits some of these differences. The chapter concludes with a discussion of genetic perturbations which have proven to be a valuable tool in studying gene regulation as well as a brief summary of some of the possible problems that may be encountered with using and measuring mRNA levels for studying gene regulation.

---

<sup>1</sup>An organism that does not possess a distinct membrane-bound nucleus

<sup>2</sup>An organism that possesses a distinct membrane-bound nucleus containing the cell's DNA



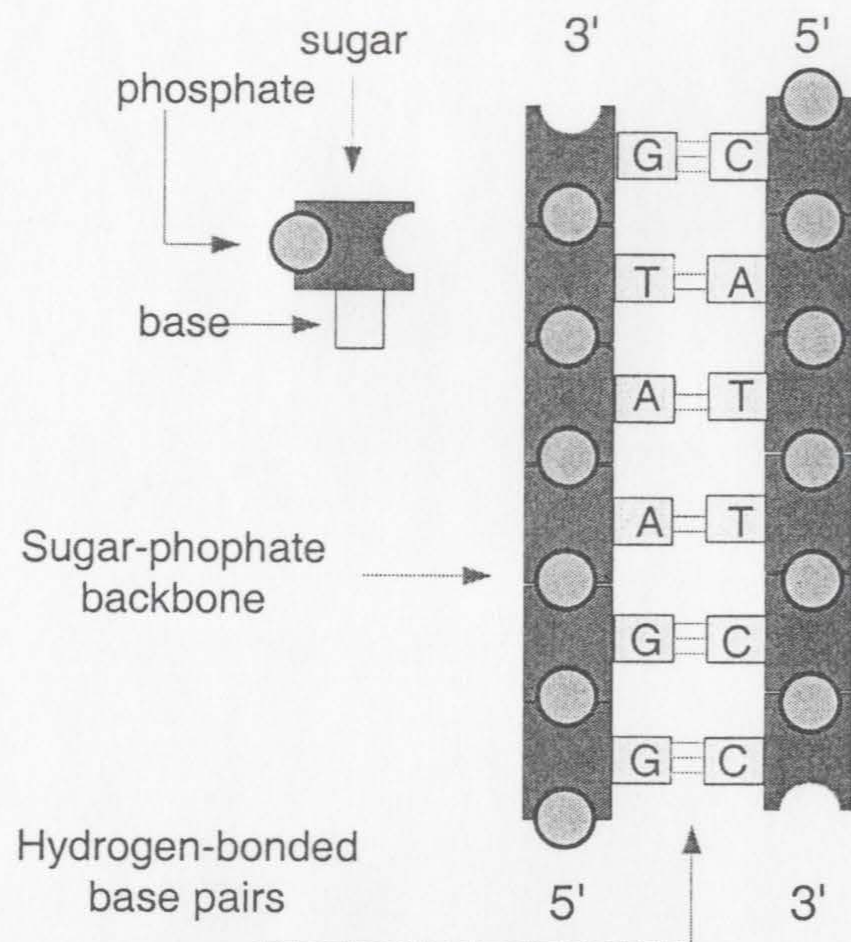


Figure 2.1: DNA ("A", "C", "G", "T") linked together in a polynucleotide chain (runs antiparallel with direction (5' end has free phosphate group, 3' end a hydroxyl group) and double helix complementary pairing (hydrogen bonds). The figure is adapted from (Alberts et al., 2002).

## 2.1 Basics of Gene Expression

### 2.1.1 DNA

The so-called blue-print of life is contained in each organism's DNA which is made up of a long string of nucleotides. A nucleotide forms a molecule of a ring compound with a nitrogen containing base linked to a five-carbon sugar (either ribose or deoxyribose) carrying one or more phosphate groups (in the case of DNA only one phosphate group is attached). These sugars form the backbone of the DNA (in the case of deoxyribose) or RNA (in the case of ribose) molecule linking each of the bases together into a long string. The phosphate group acts to bind together different units of the backbone.

If we think of each nucleotide as having a phosphate (represented by circles in Figure 2.1) and a hole, then the single string of nucleotides can be thought of to have a direction. This direction is either referred to as the  $5' \rightarrow 3'$  or  $3' \rightarrow 5'$  direction. These conventions represent the polarity of the molecule and are based on details of the chemical linkage between nucleotide subunits. There are four possible bases



in DNA, which have been assigned the letters “A”, “C”, “G” and “T” for *adenine*, *cytosine*, *guanine* and *thymine* respectively. In DNA, two strings are joined together such that their bases are paired in complementary fashion (“A” with “T” and “G” with “C”). This leads to the double helix structure commonly associated with the DNA molecule. The “G”–“C” bond is the stronger of the two with three hydrogen bonds compared with two for the “A”–“T” pair. The structure and components of a DNA molecule are shown in Figure 2.1.

### 2.1.2 The Central Dogma

The central dogma of molecular biology explains how the sequence of a strand of DNA (DeoxyriboNucleic Acid) relates to the amino acid sequence of a protein. It states that the information stored in the DNA of an organism is first transcribed into RNA (RiboNucleic Acid), which is then translated into a chain of amino acids forming a protein. Normally, it is thought that transcription and translation only proceed in the forward direction (from transcription to translation). This process is shown in Figure 2.2. However, it is known that there are exceptions to this rule. For instance, some RNA viruses can reverse transcribe themselves from RNA to DNA<sup>3 4</sup> (e.g. HIV) (Alberts et al., 2002).

Proteins can be thought of as both the workers and materials present within cells. Examples of the function of proteins include structural elements, enzymes and antibodies. Proteins can also act as transcription factors (TFs) which play an important role in the regulation of genes. These TFs bind to regulatory regions of the DNA often dramatically increasing or decreasing the subsequent transcription of nearby genes. This type of gene regulation effectively controls if and when other

---

<sup>3</sup>This reverse transcription process is also exploited by cDNA microarray technologies.

<sup>4</sup>A virus which performs this feat is known as a *retrovirus*.



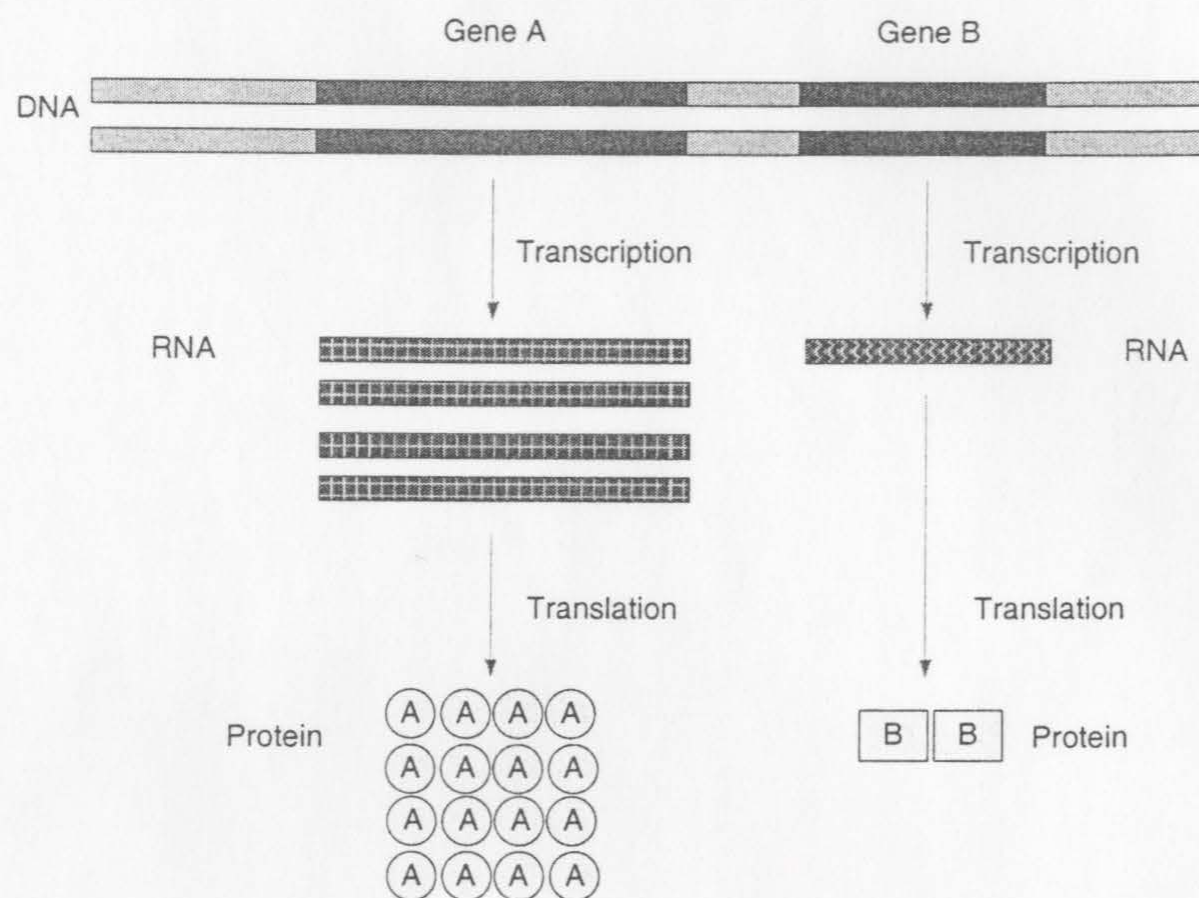


Figure 2.2: The central dogma – the transcription of DNA into RNA and the translation of RNA into proteins. In the figure, gene “A” is transcribed more efficiently than gene “B” (thus the surplus of RNA molecules compared to the number of RNA molecules generated by gene “B”). This leads to the number of “A” proteins being much larger than the number of “B” proteins. The figure is adapted from (Alberts et al., 2002).

genes are transcribed forming complex networks of interactions. These networks of interactions help lead to the sensitivity of cells to stimuli and the myriad of cellular processes, as well as to developmental stability and canalization (Siegal and Bergman, 2002).

### 2.1.3 Transcription

Transcription, the process by which a DNA molecule is used to form a molecule of RNA, is different for prokaryotes (single-celled organisms without a nucleus) and eukaryotes. As such, the process is only briefly sketched out in this section. In subsequent sections, gene regulation in both prokaryotes and eukaryotes will be separately discussed where the specifics of transcription for each cell type will be elucidated in greater detail.

As previously mentioned, transcription factors bind to regulatory regions called transcription factor binding sites. By binding to these sites, TFs can exert either



positive or negative control on the subsequent transcription of nearby genes.

The transcriptional process itself is performed by an enzyme called RNA polymerase. This enzyme is responsible for the synthesis of different kinds of RNA. RNA polymerase binds to a TF complex (the TF bound to the DNA) and DNA transcription proceeds. First, the DNA is split from its double helix form and one of the strands is read one nucleotide at a time. This strand is used as a template to produce a single stranded RNA molecule made up of four nucleotides: uracil, “U”, instead of thymine (“T”) in DNA, cytosine, “C”, adenine, “A”, and guanine, “G”. This RNA code specifies the amino acid sequence during the subsequent translation step. This RNA molecule is referred to as a messenger RNA (mRNA).

Despite small chemical differences, DNA and RNA are quite different in overall structure. Whereas DNA exists in a double helix form, RNA is typically single stranded<sup>5</sup> and thus takes on different three dimensional conformations. Because of the different conformations that RNA may take, it can also carry out many structural and catalytic cellular functions in addition to specifying the amino acid sequence of a protein. The maximum length of a molecule of RNA is typically a few thousand nucleotides, but the majority of RNAs are considerably shorter. In addition, RNA transcription is more tolerant of sequence mutations where a mutation occurs every  $10^4$  nucleotides contrasted to every  $10^7$  nucleotides during DNA transcription.

Why might a cell have an intermediate stage from DNA to the creation of a protein? Possible answers for this question are that the intermediate RNA stage buffers DNA against the caustic chemistry of the cytoplasm, that gene information can be amplified by having multiple copies of RNA from a single copy of DNA and that the additional intermediary step allows for more complex regulatory controls. Since there are more pathways inherent from the reading of the DNA to the creation

---

<sup>5</sup>Double-stranded RNA is an exception that will be discussed in Section 2.2.3.



of a given protein, there are more possibilities and pathways for control in a wider variety of environments and circumstances (Alberts et al., 2002).

## 2.1.4 Translation

Translation is the process by which a sequence of amino acids is created (forming a protein) from an mRNA molecule. Translation is initiated when an mRNA binds to a ribosome – an RNA protein complex. The ribosome then “reads” the mRNA sequence made up of the four RNA nucleotides, (“U”, “C”, “A”, “G”) which map to 20 different amino acids<sup>6</sup>. This is achieved by reading the RNA nucleotides as triplets called “codons”. This mapping of codons to amino acids is shown in Figure 2.3.

	U	C	A	G	
U	phenylalanine	serine	tyrosine	cysteine	U
	phenylalanine	serine	tyrosine	cysteine	C
	leucine	serine	punctuation	punctuation	A
	leucine	serine	punctuation	tryptophan	G
C	leucine	proline	histidine	arginine	U
	leucine	proline	histidine	arginine	C
	leucine	proline	glutamine	arginine	A
	leucine	proline	glutamine	arginine	G
A	isoleucine	threonine	asparagine	serine	U
	isoleucine	threonine	asparagine	serine	C
	isoleucine	threonine	lysine	arginine	A
	methionine	threonine	lysine	arginine	G
G	valine	alanine	aspartic acid	glycine	U
	valine	alanine	aspartic acid	glycine	C
	valine	alanine	aspartic acid	glycine	A
	valine	alanine	aspartic acid	glycine	G

Figure 2.3: Table of the mapping between the mRNA triplets composed of “U”, “C”, “A” and “G” (forming codons) and the 20 amino acids.

For instance, the triplet “UGG” codes for tryptophan. In order to ensure that the correct three nucleotides are being read as a codon, a start codon is required. When the codon “AUG” appears (coding for methionine), this signals the beginning

<sup>6</sup>There are actually 22 known amino acids. However, these 20 are sufficient to create the life we see around us.



of translation for this specific mRNA. The mRNA is translated until a stop codon is read (“UAA”, “UAG” or “UGA”). In this way, a chain of amino acids is created from an mRNA sequence. In fact, these mappings of codons to amino acids is nearly universal in nature with few exceptions. Some exceptions to this mapping include mitochondria and *Candida albicans* (a human fungal pathogen) among others (Alberts et al., 2002).

Newly created chains of amino acids then fold into a three-dimensional structure (often with the help of other proteins called “chaperones”). Typically, the products of translation are also modified by processes such as glycosylation and phosphorylation. Such post-translational modifications are thought to add functionality, affecting the targeting of proteins to certain genes, regulating activity, increasing mechanical strength, and changing the recognition of particular DNA / protein domains.

## 2.2 Models and Mechanisms of Gene Regulation

In the previous section, the process of gene transcription of DNA to an RNA molecule and subsequent translation into a protein was briefly introduced. This section describes in a little more detail how regulation of these transcriptional and translational processes can occur along with some of the differences in such processes between prokaryotes and eukaryotes.

Genetic regulation also occurs through both post-transcriptional and post-translational modifications to RNA and amino acids respectively. In fact, Day and Tuite (1998) claim that such events (at the post-transcriptional level) are the key to the successful outcomes from highly ordered developmental processes in complex organisms. Day and Tuite (1998) also provide a good overview of post-transcriptional modifications in eukaryotes. Some such modifications will be discussed subsequently. A more com-



prehensive overview of the subject matter discussed in this section can be found in Ptashne and Gann (2001) and Alberts et al. (2002).

### 2.2.1 Operon Model of Gene Regulation

The operon model of gene regulation for prokaryotes was first introduced by Jacob et al. (1960). An operon can be defined as a unit of genetic material that functions by means of an operator, a promoter and one or more structural genes. In this model, many genes are clustered together into operons and are transcribed as a single RNA transcript<sup>7</sup>.

Firstly, there are two different kinds of genes. The first type are structural genes which code for proteins. The second type are genes which code for specific RNA molecules. Specifically, it is the structural genes within an operon that are organized such that they are expressed as a single mRNA. Expression of this mRNA depends on the presence or absence of a regulatory protein acting as an inhibitor or activator. The second type of gene codes for these regulatory proteins which function to regulate the expression of other genes. These act on other DNA molecules, and are therefore referred to as *trans*-acting<sup>8</sup> factors (usually proteins).

The initiation of transcription is controlled by two DNA sequence elements (called promoters) approximately 35 bases<sup>9</sup> and 10 bases<sup>10</sup> (often referred to as the Pribnow-box) upstream of the transcriptional initiation site. The DNA site to which a regulatory protein (usually acting as a repressor in prokaryotes) binds is referred to as an operator. The operator site is usually located adjacent to the promoter.

In the operon model, there are two major modes of transcriptional regulation:

---

<sup>7</sup>These RNAs are referred to as polycistronic.

<sup>8</sup>*trans* refers to a factor which affects DNA molecules other than the one from which it was created. The DNA molecules that the *trans*-acting factor act upon are referred to as *cis*-acting.

<sup>9</sup>The consensus sequence of this promoter is "TTGACA".

<sup>10</sup>The consensus sequence of this promoter is "TATAAT".



catabolite-regulated operons (gene products necessary for the utilization of energy) and attenuated operons (gene products necessary for the synthesis of small biomolecules such as amino acids). The operon model, along with these two modes of transcriptional regulation are best illustrated by the examples of the *lac* and *trp* operons.

### The *lac* operon

An example of a catabolite-regulated operon is the *lac* operon. The *lac* operon is responsible for regulating the metabolism of lactose – an energy source. Since the use of the sugar glucose is more energy efficient than lactose, lactose is typically only used in the absence of glucose.

There are four genes related to the *lac* operon, one regulatory gene, *lacI*, for lactose inhibitor, and three structural genes, *lacZ*, which codes for  $\beta$ -galactosidase (which cleaves lactose as the first step in metabolism), *lacY*, which codes for  $\beta$ -galactoside permease (which increases the permeability of the cell to  $\beta$ -galactosides), and *lacA*, which codes for  $\beta$ -galactoside transacetylase as shown in Figure 2.4.

RNA polymerase (RNAP) binds with the promoter (which is approximately 60 base pairs in length) on the left of Figure 2.4 transcribing the *lacI* gene. The protein product of the *lacI* gene forms a tetramer which binds to the operator (which is approximately 20 base pairs in length) shown in the figure. When the tetramer is bound to the operator, RNA polymerase is prevented from binding to the adjacent promoter site. This effectively prevents the transcription of the three structural genes, *lacZ*, *lacY* and *lacA* preventing the use of lactose as an energy source. When these genes are inhibited, lactose cannot be used as an energy source even if it is present within the cell (assuming that glucose is present). This is called catabolite repression.

However, if an inducer is present (such as allolactose), the conformation or shape of the tetramer produced by the transcription of *lacI* changes. This change is such that



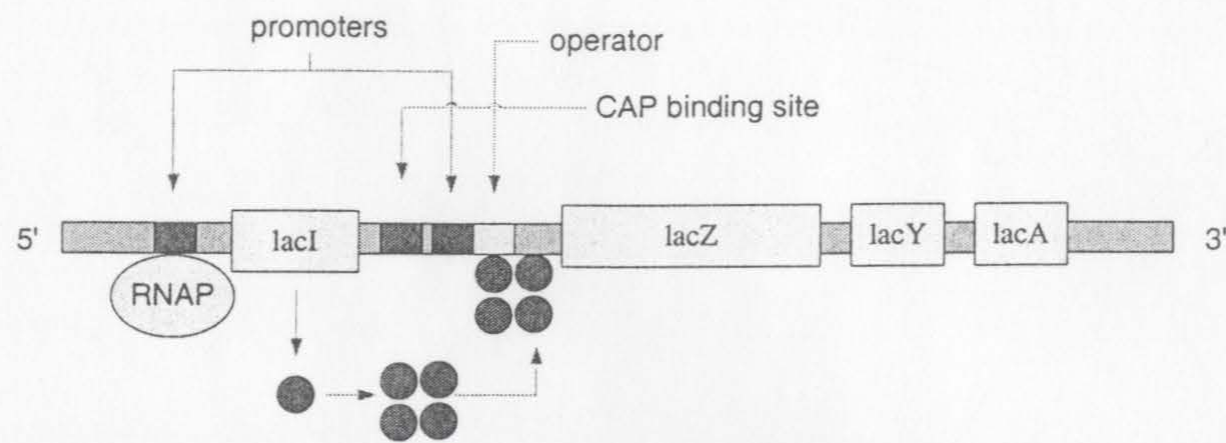


Figure 2.4: The lactose operon. RNA polymerase attaches to the leftmost promoter transcribing the *lacI* gene. However, the protein product of *lacI* forms a tetramer which attaches to the operator site preventing RNA polymerase from binding with the rightmost promoter thus preventing transcription of *lacZ*, *lacY* and *lacA*.

the tetramer can no longer bind with the operator site. This removes the impediment to transcription that was previously described. Therefore, the *lacZ*, *lacY* and *lacA* genes are now transcribed allowing the metabolism of the lactose sugar. This is shown in Figure 2.5.

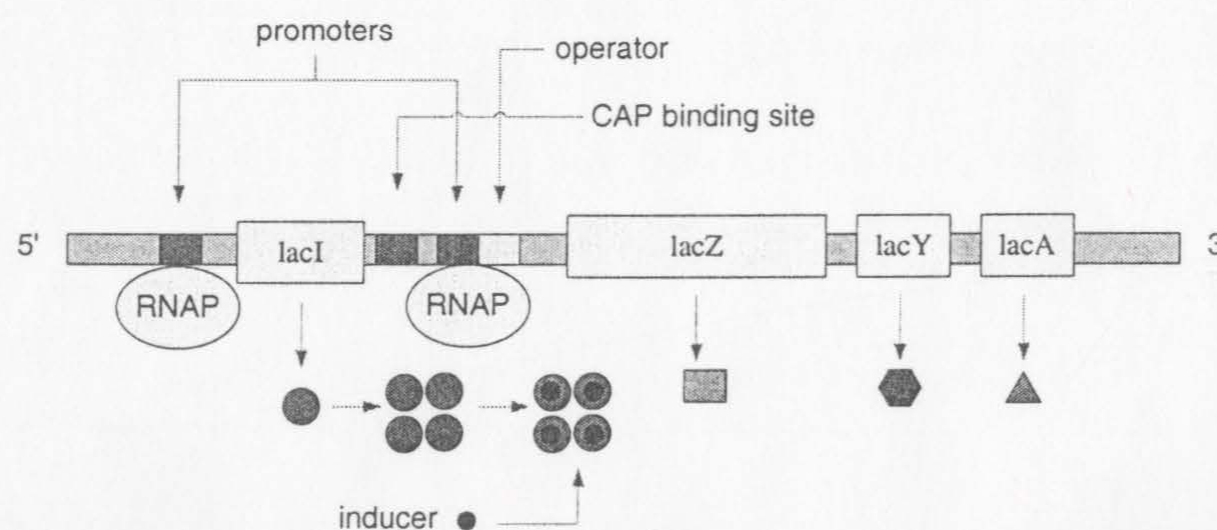


Figure 2.5: The lactose operon. RNA polymerase attaches to both promoters transcribing the *lacI*, *lacZ*, *lacY* and *lacA* genes. The protein product of *lacI* is prevented from binding with the operator by an “inducer” which changes the conformation of the tetramer preventing it from binding. Thus, transcription of the last three genes, *lacZ*, *lacY* and *lacA* proceeds.

The *lac* operon may also be positively regulated in the absence of glucose through binding of the cAMP–receptor protein to sequences near the promoter domain of the operon called the CAP site (which is approximately 20 base pairs in length). This



results in additional recruitment of RNA polymerase to the promoter. This type of activation results in a 40 to 50-fold increase in polymerase activity. This form of regulation is what makes the *lac* operon catabolite-regulated. There are also two additional operator sites where the repressor tetramer binds. These sites are located 90 base pairs upstream and 400 base pairs downstream and bind co-operatively with the operator site by looping of the DNA.

### The *trp* operon

The *trp* operon is an example of an attenuated operon. The *trp* operon controls the production of tryptophan. It functions much in the same way as the previously presented *lac* operon. There is a repressor gene (*trpR*) whose protein product binds to an operator site. One small difference is that instead of forming a tetramer as was the case with the protein product of *lacR*, *trpR* forms a protein dimer. However, the main difference is that the *trpR* protein dimer cannot by itself bind to the *trp* operator. Therefore, the five structural genes of the *trp* operon are normally transcribed. So if the *trpR* protein dimer cannot bind to the operator, how does it regulate the operon? This is accomplished with the aid of two tryptophan molecules which bind to the *trpR* protein dimer. This binding effectively changes the molecular conformation of the molecule allowing it to bind to the operator thereby halting transcription of the *trpE*, *trpD*, *trpC*, *trpB* and *trpA* structural genes. This process is shown in Figure 2.6.

Therefore, this form of repression only occurs when there is sufficient tryptophan present. When the concentration of tryptophan is high, there is significant binding of tryptophan with the *trpR* protein dimer which shuts off further production of tryptophan. As the concentration of tryptophan falls, there comes a point where there is not a sufficient number of tryptophan molecules present to bind with the



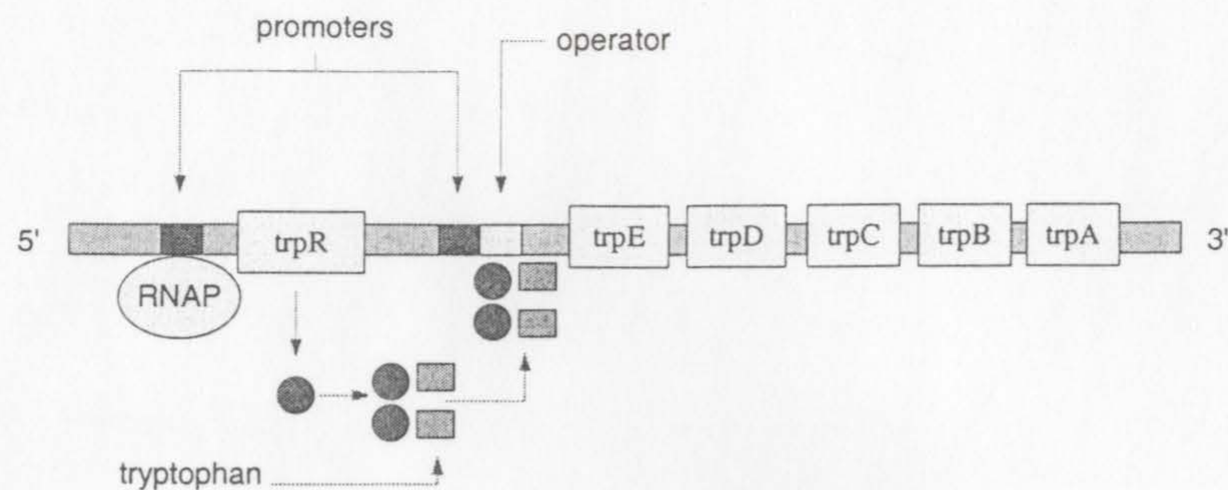


Figure 2.6: The tryptophan operon. RNA polymerase attaches to the leftmost promoter transcribing the *trpR* gene. The dimer protein product of *trpR* cannot attach to the operator site to prevent binding by RNA polymerase without two molecules of tryptophan. The combined dimer / tryptophan complex binds to the operator preventing the transcription of *trpE*, *trpD*, *trpC*, *trpB* and *trpA*.

*trpR* protein dimer. When this occurs, the tryptophan / *trpR* protein complex is unable to bind with the operator and transcription again resumes. In this way, the production of tryptophan is regulated such that its production is halted only when the concentration is too high. This kind of gene regulation of an operon is referred to as an attenuated operon.

### 2.2.2 Control of Gene Expression in Eukaryotes

The previous section illustrated how gene expression can be regulated in prokaryotes. However, the regulation of gene expression in eukaryotes is more complex. In particular, higher eukaryotes (multi-cellular eukaryotes) need to regulate their genes for cell specialization. Each cell type comes into being by differentially activating a different subset of genes during development. Therefore, with much more complex regulatory interactions required in order to direct the development of a multitude of cell types, it is not surprising that the control of eukaryotic gene expression is more complex than in prokaryotes.

However, before describing some of the mechanisms of regulation for eukaryotes,



some of the general differences between prokaryotes and eukaryotes are discussed. As was previously mentioned, eukaryotic organisms possess a nucleus – a spherical membrane within the cell which contains the cell's DNA. Genetic transcription occurs within a eukaryotic cell nucleus with mRNAs having to traverse the nucleus membrane in order to be translated into proteins. In contrast, in prokaryotes, transcription and translation both occur throughout the cell.

In terms of regulatory regions, in prokaryotes the regulatory region is usually directly upstream of the coding region. But in eukaryotes, the regulatory region could be a considerable distance up or downstream of the coding region. In fact, gene regulatory proteins can act thousands of nucleotides away from the promoter they influence. This sort of “action at a distance” makes regulatory relationships much more complex. Many eukaryotic genes also have one or more enhancers (quite a distance away from the coding region) responsible for regulating cell- or tissue-specific transcription (the transcription responsible for cell differentiation).

Eukaryotic promoters contain the so-called “TATA” box (25 nucleotides upstream of the initiation site of transcription). In addition, in contrast to the operon model where many structural genes are transcribed into a single mRNA, each eukaryotic gene typically requires its own promoter (no operons). Another key difference is that prokaryotic genes are most often regulated by repressors, while eukaryotic genes are primarily regulated by transcriptional activators.

In addition, eukaryotic transcription requires many additional proteins, some of which are called general transcription factors. In fact, there are three different kinds of RNA polymerase in eukaryotes as opposed to only one in prokaryotes. Because of these differences there are also more opportunities for regulatory relationships.

For eukaryotes, DNA is also compacted into chromatin, which affects the ability of transcription factors and RNA polymerase to find access to genes. Thus, this pack-



aging of DNA in chromatin allows for additional regulatory opportunities. A number of modifications to RNA (some for additional stability) are possible such as capping, addition of a poly-A tail, methylation, cleavage of big RNAs and splicing (Alberts et al., 2002). Splicing plays an important role in eukaryotic cells. In eukaryotes, not all of the pre-mRNA which is transcribed from the DNA becomes translated into a chain of amino acids. An intermediate step occurs called splicing. Parts of the pre-mRNA, called introns, are removed from the pre-mRNA sequence leaving portions referred to as exons. This process is illustrated in Figure 2.7.

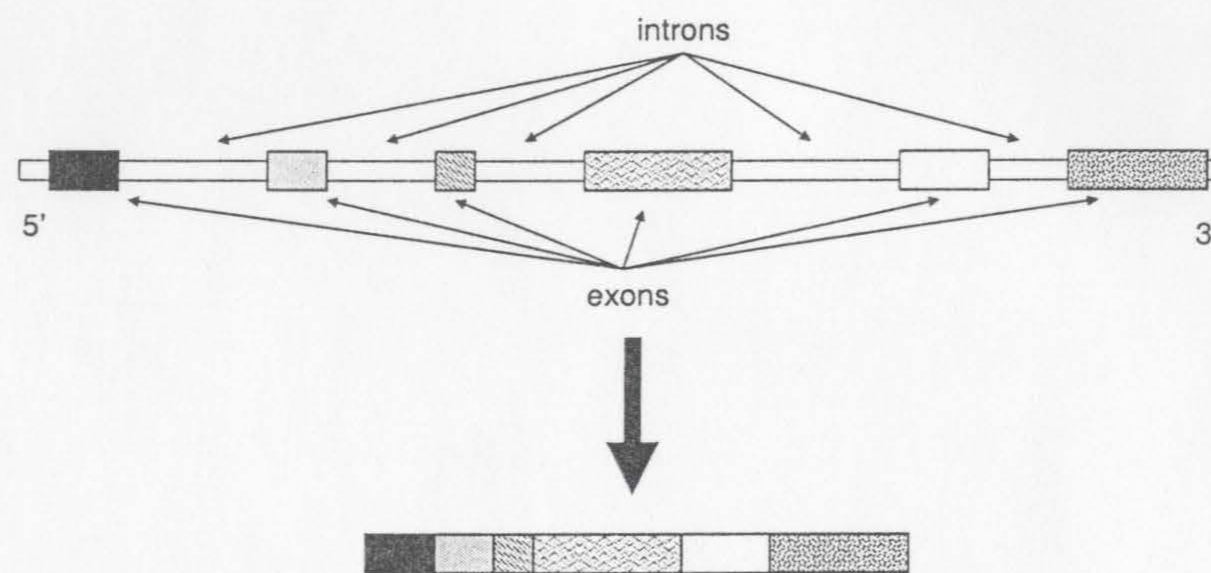


Figure 2.7: A single strand of DNA is shown above. The resultant mRNA after the removal of introns leaving behind the exons is shown below. After additional post-transcriptional modifications, this mRNA will be translated into a chain of amino acids.

Since splicing can occur at different sites for a given gene, this allows for the creation of different mRNA transcripts from the same gene. There are four known modes of alternative splicing:

1. Alternative selection of promoters – A different promoter is spliced with a given set of exons.
2. Alternative selection of cleavage / polyadenylation sites – Different polyadeny-



lation sites are spliced with other exons

3. Intron retaining mode – An intron is retained in the mRNA transcript
4. Exon cassette mode – A given exon is spliced out of the mRNA

It has been hypothesized that alternative splicing possibly allows for faster evolution, and a higher coding efficiency since several proteins can be encoded in a sequence whose length would only be enough for two proteins if coded in the same ways as for prokaryotes.

Overall, the main regulatory mechanisms or processes where regulation can be performed in the eukaryotic cell are (Ptashne and Gann, 2001, Alberts et al., 2002):

1. Transcriptional Initiation – There exist differential strengths of promoter elements in addition to the presence / absence of activator sequences (which enhance RNA polymerase activity). These can dramatically affect the number of transcripts produced for a given gene.
2. Transcript Processing and Modification – Eukaryotic mRNA needs to be capped, polyadenylated, and have its introns removed. Some genes undergo tissue-specific alternative splicing (generate biologically different proteins).
3. RNA transport – In eukaryotes, mRNA must leave the nucleus to be translated.
4. Transcript Stability – Prokaryotic transcripts remain stable for about one to five minutes. Eukaryotic transcripts have a much greater variability in their lifetimes.
5. Translational Initiation – Expression of a gene product is affected by the ribosomes ability to recognize and initiate translation from the correct methionine



("AUG") codon since many genes have multiple methionine codons. The translation of some eukaryotic proteins can also be initiated at non-"AUG" codons (Alberts et al., 2002).

6. Post-Translational Modification – Common modifications after the translation step include glycosylation, acetylation, fatty acetylation, and the formation of disulfide bonds.
7. Protein Transport – In order for proteins to be effective as enzymes or as building blocks, they must be transported to the site of action.
8. Control of Protein Stability – Many proteins quickly become unstable and degrade whereas others are highly stable and have significantly longer lifetimes. It has been shown that some specific amino acid sequences bring about rapid degradation in proteins.

### 2.2.3 RNA interference (RNAi) and microRNA (miRNA)

Although the gene activation and inhibition mechanisms previously presented are typically the primary means by which genes regulate the expression of other genes, there also exist other mechanisms. Two such mechanisms are through the RNA interference effect and the existence of miRNA which are both discussed in this section. These mechanisms may also be used in genetic perturbation experiments which are discussed in the subsequent section.

When an RNA sequence complementary to an mRNA (anti-sense RNA) is injected into a cell, the two sequences hybridize forming a double-stranded RNA (dsRNA) duplex thus blocking translation of the mRNA into a protein (since a ribosome only binds to a single-stranded RNA).



However, through studies on the nematode, *Caenorhabditis elegans*, it was discovered that injection of the sense and anti-sense strands at the same time led to a higher level of gene silencing than when injecting either type of strand on its own (Fire et al., 1998) – a somewhat counterintuitive result. In fact, injection of only a few molecules of dsRNA served to completely silence the expression of the homologous gene. This effect is known as the RNA interference (RNAi) effect. In addition, injection of dsRNA in the gut of *Caenorhabditis elegans* also effectively silenced expression of the homologous gene even in the organism's first generation of offspring (Fire et al., 1998). Thus, use of RNAi is one potent method for silencing the expression of a given gene.

The RNAi effect is produced by the presence of dsRNA which triggers the natural degradation of the homologous mRNA (called nonsense-mediated decay). This mechanism is used to prevent the production of defective protein molecules from mRNA. In practice, RNAi is produced by the injection of small interfering RNAs (siRNAs) which form 21–25 nucleotide dsRNAs. It is believed that dsRNA acts as a catalyst by forming an RNA-induced Silencing Complex (RISC) where the unwound siRNAs base pairs with complementary mRNA. This then guides the RNAi machinery to target mRNA resulting in the effective cleavage and subsequent degradation of the mRNA. The activated RISC could then potentially target other copies of the mRNAs functioning as a catalyst. More details of the specific molecular mechanisms responsible for the RNAi effect can be found in Hammond et al. (2001). Since the homologous mRNA is quickly degraded, the translation of mRNA into protein is effectively halted. However, only dsRNA which targets exon sequences is effective in producing an interference effect. dsRNAs which target promoter and intron sequences do not appear to produce any RNAi effect (Fire et al., 1998).

The use of RNAi is currently becoming an important method with which to probe



the workings of genetic network interactions and is discussed in Skipper (2003) and Section 2.3. It is emphasized that it is the presence of dsRNA, not the single-stranded anti-sense RNA which produces the interference effect. In addition, the RNAi effect is highly specific and remarkably potent (only a few dsRNA molecules are required to produce the interference effect). This effect can cause interference in cells and tissues far removed from the original site of injection and even in subsequent offspring (Fire et al., 1998).

A gene silencing (repressive) effect similar to that observed in RNAi has also been observed with microRNAs (miRNAs) in eukaryotic gene expression (Carrington and Ambros, 2003). miRNAs are translated as parts of longer RNA molecules and are processed by dsRNAs in the nucleus resulting in 19–23 mer miRNAs. These miRNAs are bound to a complex that participates in RNAi. This complex can bind to sequences that are significantly similar but not completely complementary to the corresponding mRNA. However, in contrast to RNAi which involves RNA degradation, the bound mRNA complex simply remains untranslated reducing the expression of the corresponding gene. A full characterization of the molecular underpinnings of this process are currently unavailable (Carrington and Ambros, 2003). miRNAs influence a variety of processes including early development (Reinhart et al., 2000), cell proliferation and cell death (Brennecke et al., 2003), and apoptosis and fat metabolism (Xu et al., 2003).

## 2.3 Genetic Perturbation Experiments

Although gene perturbations such as gene knockout (deletion) and over-expression are not a specific technology per-se, they are an important method for probing the workings of genetic regulatory networks. Such methods function by introducing a



perturbation into the network with subsequent evaluation of the steady-state expression level of all genes in the network in the presence of this perturbation. At the least, it can be determined whether or not a given gene is crucial to the organism's survival. In fact, it was through gene perturbation analysis that the knowledge that most genes are pleiotropic (expressed in different tissues in different ways at different times) was discovered. From this data, it is also possible to infer portions of the regulatory interactions among genes. As such, these concepts have been previously applied to inferring the structure of regulatory networks (Ideker et al., 2000, Wagner, 2001, 2002, Gardner et al., 2003, Tegnér et al., 2003, Bongard and Lipson, 2004, Bernardo et al., 2004).

In perturbing a genetic system, the perturbations can be either genetic or biological (i.e. a non-genetic factor is altered such as a change in growth media, temperature, or addition of an extracellular ligand). In this section, only the former types of perturbations (i.e. perturbations at the genetic level) are discussed.

One such perturbation is gene knockout. The effect of the absence of a given gene in the network can both lead to explanations of the gene's role, as well as elucidation of the roles of other genes that would normally be co-expressed. Knocking out a gene in a given organism may be accomplished in a variety of ways. One typical method (used in mice) is to engineer DNA which has a mutant copy (effectively making it non-functional) of the gene to be knocked out. This DNA is then introduced into special embryonic stem cells. If the foreign introduced DNA is similar in sequence to the host DNA, it may undergo "homologous recombination" (it forms a single copy of the sequence at the specific site).

Cells in which the mutant gene has indeed replaced the native copy are then introduced into early embryos. After mice with this form of genetic manipulation are born, they are mated to each other. Since each individual mouse contains only one



mutant copy of the gene in their DNA (mice are diploid organisms), any offspring of such a pairing have a one in four chance of possessing two mutant copies of the gene. It is these individuals who have effectively had a specific gene knocked out since no good copies of the original gene exist in these individuals. Such individuals are referred to as being “transgenic”. A more complete review of the technology involved in gene-knockout experiments can be found in Galli-Taliadoros et al. (1995). Gene deletion data has been obtained for a wide variety of model organisms such as *Saccharomyces cerevisiae* by a world-wide academic consortium which generates and collects various mutant strains which have a specific gene deleted (Winzeler et al., 1999).

In addition to perturbation by gene deletion, gene over-expression can also be achieved. There are primarily two major methods for creating gene over-expression: the introduction of foreign DNA into embryonic stem cells (similar to what was previously described for gene deletion) and “pronuclear microinjection”. The former method has already been discussed and is achieved similarly to gene deletion. In the case of pronuclear microinjection in mice, the foreign DNA is introduced into the mouse egg immediately after fertilization via a fine needle into the male pronucleus (derived from the sperm). This DNA tends to integrate as tandemly arranged copies randomly into the genome. Once again, only some of the cells produced will be transgenic, thus making the specific organism only partly transgenic. Obtaining a fully transgenic individual requires the screening and mating of two partially transgenic individuals.

Another tool with which to perturb the functioning of genes *in vivo* is through the use of double-stranded RNA (dsRNA) and the RNA interference (RNAi) effect. An experimenter could use dsRNA to perturb the network such that a given gene is silenced. The resultant network can then be analyzed and compared to the dynamics of the unperturbed network in order to determine the silenced gene’s role in the



network's dynamics. However, although small interfering RNAs (siRNA) in a virus-infected cell did silence their respective genes, the siRNAs also activated the cell's interferon target genes. These findings are summarized in Skipper (2003). This would seem to indicate that the results of RNAi-type experiments must be interpreted with care.

## 2.4 Caveat Emptor with mRNA Measurements

In order to determine the relationships between the underlying genes and proteins that form a regulatory network, data is required. Typically, this comes in the form of the measurement of the expression of mRNA which is often the first step in any study of genetic regulatory networks. By measuring the expression of given mRNAs, it can be inferred how frequently and in what quantity mRNA is being generated from associated genes. From these patterns of expression, a network of interactions between genes and their protein products can be inferred through subsequent analysis. Some typical high throughput technologies used for this purpose are genetic microarrays, serial analysis of gene expression (SAGE), chromatin immunoprecipitation (ChIP), and mass spectrometry. A brief description of some of these methods is presented in Appendix A.

Although the measurement of mRNA is a powerful method with which to probe the workings of regulatory networks, the caveats of such an approach must be realized (Kitano, 2001). Even at equilibrium, different mRNAs are degraded at specific rates as a function of the rate of transcription, stability of the mRNA molecule, and changes in processing due to other cellular events which can be either internal or external. Therefore, this can lead to completely divergent mRNA measurements (e.g. when two genes are transcribed simultaneously and at the same rate, but whose mRNAs



have different decay rates).

In addition, because of the different decay rates, even the presence of a molecule of mRNA does not mean it was recently transcribed if its decay rate is very slow. Also, the time scales at which transcription and translation occur can be of greatly different magnitudes. mRNA can be transcribed up to several hundred nucleotides per minute, but can also take a significantly longer time (e.g. the gene dystrophin which is found in muscle takes 16 hours to transcribe (Tennyson et al., 1995)). Also, the implicit assumption when measuring mRNA levels is that they are translated into proteins at an equal rate. Even if this were true, the life span of proteins is at least as variable as mRNA. Therefore, gene expression activity may not always be a good indicator of corresponding concentration or activity of a given protein. Furthermore, even if an mRNA is found to be expressed, it does not imply that it will be translated into a protein at all (for example by the process of retroregulation).

Other difficulties also exist for mRNA technologies which are too specific. In eukaryotes, mRNAs generated by transcription may differ depending on the incorporation of different exons and introns (alternative splicing). If the technology being used only measures one of these products, a misleading or incomplete picture of the transcription occurring in the organism may be obtained. In fact, the genetic basis for organismal diversity is often attributed to polymorphisms of each gene – so-called single nucleotide polymorphisms (SNPs). However, only some SNPs result in different proteins. If expression measurements only look for a single type of SNP, an inaccurate picture of the transcription of the organism will again be obtained.



## 2.5 Conclusion

The description of the biology behind gene expression presented here is merely a caricature which outlines some of the main fundamental concepts. It is in no way a complete picture of the processes inherent in transcription and translation. For a more complete picture, please refer to Davidson (2001), Ptashne and Gann (2001) and Alberts et al. (2002).

The processes and mechanisms described here are all subject to the forces of selection and evolution. Therefore, an understanding of the possible evolutionary processes that have shaped genomes and their related processes can be useful in studying gene regulatory interactions such as gene duplication (Ohno, 1970, Zhang, 2004) and lateral gene transfer in prokaryotes (Ochman et al., 2000, Woese, 2002). A brief review of gene duplication and its potential role in shaping regulatory interactions will be presented in Section 5.1. A more comprehensive view of the evolution of prokaryotes and eukaryotes can be found in Maynard Smith (1998).

Having introduced the basics of how genes regulate each other, it is not hard to imagine large networks of interacting genes and proteins which regulate each other. Such interactions produce the complex organisms, morphologies, and cell types we see in nature. Modelling these types of interactions in an abstract manner is the subject of the next chapter.



## Chapter 3

# Network Models

The study of the organization and functionality of gene regulation has important consequences for our understanding of the basics of development and evolution. With the desire to study regulatory relationships on a larger scale, the need for formalisms arises. The choice of formalism is often problem and domain dependent. For instance, is the goal to forward model or reverse-engineer a system? Forward modelling is accomplished by taking what is currently known about a system and formulating a model to see where the model agrees and disagrees with the natural system. In this sense, forward modelling is a form of knowledge-driven model construction. Reverse-modelling or reverse-engineering tries to use the behaviour of the system itself to directly infer the interactions of the natural system. These interactions are then formulated into a model. Thus, the formulation of a model generated by reverse-engineering is data-driven.

This section introduces many of the most common formalisms including those based on Boolean networks, differential equations, and stochastic models. Many of these formalisms have been used for both the forward and reverse-modelling problems. In particular, an artificial regulatory network model by Banzhaf (2003a) is introduced



which is more extensively studied subsequently in this thesis.

It should be noted that many of the networks modelled using these formalisms often first use a clustering step in order to reduce the complexity of the problem domain. Without such a reduction in complexity, some of the analysis methods presented become intractable for certain problem domains. Some of the more common methods for clustering and projection of gene expression data are presented in Appendix B.

### 3.1 Boolean Network Models

The Boolean network model was introduced by Kauffman (1969a,b), Glass and Kauffman (1973) and Kauffman (1974). In this model, genes are represented by nodes whose states are either “0” or “1”. This means that a gene is either being transcribed, or it is not. There are no intermediate levels of transcription. Each node is connected to others by directed edges as shown in Figure 3.1(a). The binary values from other nodes are received through the incoming edges and taken as inputs. These are sent through a Boolean function that determines the current state of the node. Typical Boolean network models use a synchronous updating scheme where each node in the system is updated at the same time during each time step. This synchronous updating scheme simplifies the simulation and analysis of Boolean networks, while purportedly preserving the network’s qualitative generic dynamics (Wuensche, 1998).

Boolean networks are often specified as  $(N, K)$ -networks where  $N$  specifies the total number of nodes in the system and  $K$  specifies the maximum number of incoming edges for each node. Typically, the value of  $K$  is small (often three) meaning that only a small subset of all possible genes is responsible for controlling the activity of any given single gene. Such an assumption can be problematic since it is known that many portions of transcriptional regulatory networks display scale-free connectivity



indicating that there are genes which have a large number of transcriptional regulators (this is discussed in more detail in Chapter 4).

An example of a Boolean network is shown in Figure 3.1(a). The connectivity and the Boolean update functions for the network are given in Figure 3.1(b). This example has five genes,  $N = 5$ , with maximum connectivity,  $K = 3$ .

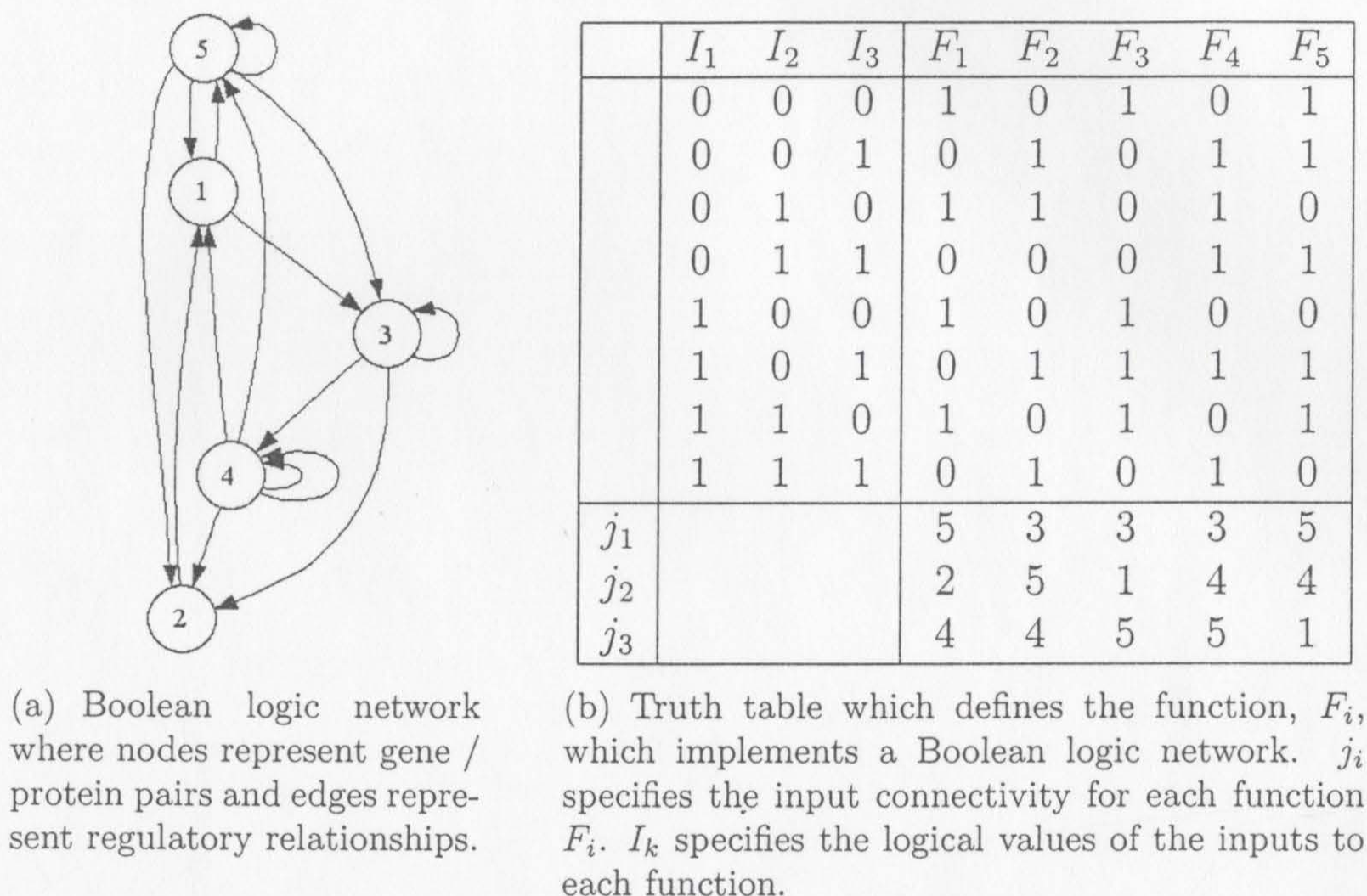


Figure 3.1: An example of a Boolean network.

The dynamics of this network are shown in Figure 3.2. Since there are five genes in the network there are  $2^5 = 32$  different states. Each state is represented by a circle in the figure while the directed edges show the state transitions of the network according to the functions in Figure 3.1(b).

Notice in Figure 3.2 that there are two limit cycles<sup>1</sup>. The first occurs between the 10110 and 01101 states. The second limit cycle involves the following states: 10101  $\rightarrow$  10011  $\rightarrow$  00010  $\rightarrow$  01110  $\rightarrow$  01100. There are also several states which

<sup>1</sup>a sequence of repeating states



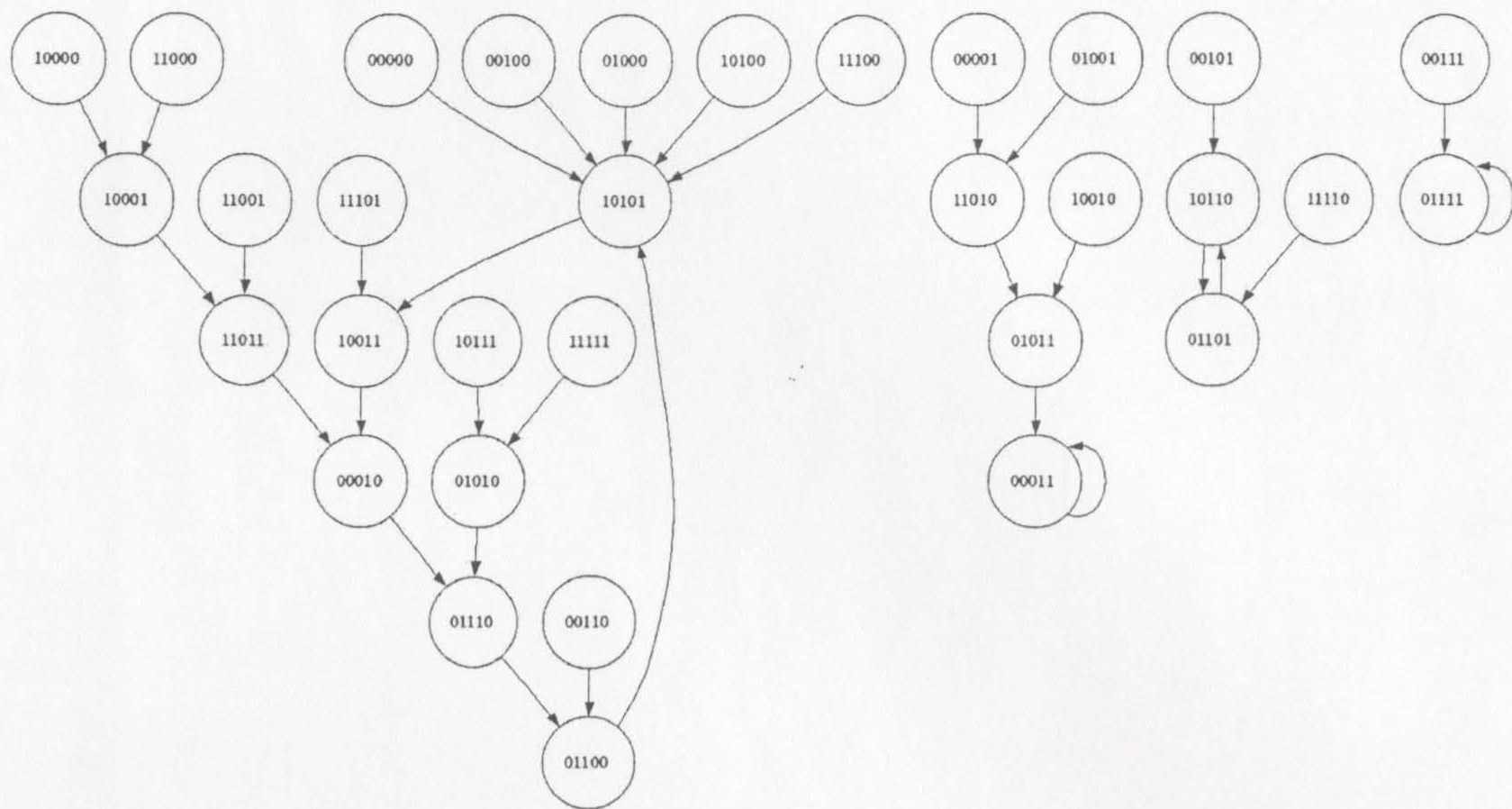


Figure 3.2: The state-transition diagram for the Boolean network defined. There are two limit cycles and two stable states.

move onto these limit cycles. There are also two stable states, 01111 and 00011.

Liang et al. (1998), Akutsu (1999), Akutsu et al. (2000b), D’Haeseleer et al. (2000), Silvescu and Honavar (2001) and Albert (2004) have used the Boolean network framework in both the forward modelling and reverse-engineering of genetic regulatory networks. However, many have questioned the “all or nothing” level of gene transcription in the Boolean model (Glass and Kauffman, 1973, Thomas and Kaufman, 2004a,b). It has been noted that there are many genes which have different regulatory effects depending on their level of expression. In addition, it is thought that the transient period between when a gene switches may also be significant. It has also been suggested that there is not a direct correspondence between the dynamic behaviour of Boolean systems and their continuous counterparts indicating a qualitative loss of behavioural information (Glass and Kauffman, 1973).

The assumption of synchronous updates of gene states may also be problematic. For instance, if gene “*a*” crosses its threshold instantaneously before gene “*b*”, the



resultant state of a Boolean model would be different from the case where “*b*” crossed instantaneously before “*a*”. Attempts to use asynchronous updating behaviours have also shown that the updating scheme may interfere with many of the interesting phenomena displayed by Boolean networks (Harvey and Bossamaier, 1997).

More generalized logical models which work on asynchronous updating schemes and multiple logical thresholds have also been used primarily in forward modelling applications (Thomas and Kaufman, 2004a,b). In addition, Mestl et al. (1997), Edwards (2000), Edwards and Glass (2000), Edwards (2001), Edwards et al. (2001) and Ben-Hur and Siegelman (2004) have introduced a differential equation framework which combines many of the advantages of Boolean networks with differential equation models. This work is presented in Section 3.2.3.

## 3.2 Differential Equation Models

Differential equation models offer an alternative to the Boolean models previously described. Such models have been used extensively in the sciences to model a variety of systems (Voit, 2000, Zill, 2000, Bower and Bolouri, 2001). Since such models are based on sets of differential equations, there are no problems with the synchronous updating of variables since the solutions obtained are inherently asynchronous. In addition, the use of differential equations removes the assumption of the binary expression of genes and proteins thereby allowing quantitative as well as qualitative comparison between computational models and the systems they are trying to model.

However, differential equation models often contain many parameters which must be obtained from the literature, by experiment or by guesswork. In addition, for some of the more complex differential equation formalisms, analytical solution and analysis of the equations may be impractical even for small network models.



### 3.2.1 Ordinary Differential Equations

The basis for much of the work in modelling regulatory networks using ordinary differential equations comes from the chemical rate equations from which the Michaelis–Menten equations are derived (Voit, 2000). The rate law for a substrate,  $S$ , and a product,  $P$ , is:

$$\dot{S} = -\frac{V_{max1}S}{K_{M1} + S} \quad \dot{P} = \frac{V_{max2}S}{K_{M2} + S} \quad (3.1)$$

Equation 3.1 explicitly models the concentration of a protein and gene pair and are known as the Michaelis–Menten equations. The two parameters,  $V_{max}$  and  $K_M$ , control the magnitude of the rate of change of the substrate.

This formalism may also be generalized as follows:

$$\frac{dr_i}{dt} = f_i(\mathbf{p}) \quad \frac{dp_i}{dt} = g_i(\mathbf{r}) \quad (3.2)$$

where  $\mathbf{p}$  represents a vector of protein concentrations and  $\mathbf{r}$  represents a vector of mRNA concentrations, and the functions,  $f_i$  and  $g_i$ , represent updating functions. Typical functions for  $f_i$  and  $g_i$  are sigmoidal in shape. One of the most commonly used functions are Hill functions of the form  $f(x_j, \theta_{ij}, n) = \frac{x_j^n}{x_j^n + \theta_{ij}^n}$ .

### 3.2.2 Weight Matrices

Weight matrices attempt to model regulatory networks with the use of linear coefficients representing the relationships between genes (Weaver et al., 1999). Thus, an individual gene's expression can be determined by the mathematical summation of all



its independent regulatory inputs (multiplied by their respective coefficients). Such a scheme is often represented in the form of a matrix where an entry in the matrix  $(i,j)$ , represents the effect of gene  $i$  on gene  $j$ . Therefore, the total regulatory input to gene  $i$ ,  $r_i(t)$ , is found as follows:

$$r_i(t) = \sum_j w_{i,j} u_j(t) \quad (3.3)$$

where  $w_{i,j}$  gives the effect of gene  $i$  on gene  $j$  and  $u_j(t)$  gives the expression state of gene  $j$  at time  $t$ . Positive values of  $w_{i,j}$  lead to activation while negative values lead to repression. A value of zero indicates no interaction.

If we take the matrix  $M(i,j)$  to give the expression of gene  $i$  at time  $j$ , a vector “ $a$ ” corresponds to the transpose of the weight matrix row of the gene of interest, and a vector “ $b$ ” corresponds to the relative expression level of the gene of interest at the given state transition. This system of equations,  $Ma = b$ , may then be solved. Since there are often more genes than data points, this problem is under-determined as there are many solutions that satisfy the above equation which may be problematic.

There are numerous assumptions when modelling regulatory networks using weight matrices. One such assumption is that all genetic interactions may be treated as independent events which is known to be false (Weaver et al., 1999). As well, transcription must be assumed to be a discrete time system in order to make the problem computationally tractable. Also, the weights assume a linear relationship between the number of copies of a gene’s mRNA and the amount of produced gene present in the cell. It is also assumed that a gene’s “maximal” expression level can be determined by empirical observation. When genes are being expressed near the maximum or minimum levels it becomes difficult to have useful predictions of the input regulatory



state due to the normally sigmoidal nature of the assumed dose-response curve (the use of other curves is possible, but this problem is still present regardless of the curve used). Small noise perturbations can exacerbate this problem leading to progressively larger errors in the regulatory state calculation.

One of the benefits of modelling regulatory networks in this fashion is that mathematical approaches found in linear algebra can be used for analysis of the resultant models (Weaver et al., 1999). However, since the matrix values (the effect of a gene on other genes in the network) are not known in advance, they must be deduced often using statistical methods or machine learning techniques such as simulated annealing, neural networks or genetic algorithms (Ando and Iba, 2000, 2001).

### 3.2.3 Piece-wise Linear Differential Equation Model

One approach which combines the logical rules of Boolean network models with some of the advantages of differential equation methods are “Glass networks”. Glass networks have been proposed as a simplified model of genetic networks (Edwards and Glass, 2000, Edwards, 2001, Edwards et al., 2001, Ben-Hur and Siegelman, 2004, de Jong et al., 2004, Mason et al., 2004) as well as an underlying model for the reverse-engineering of regulatory networks (Perkins et al., 2004) and to model neural networks (Edwards, 2001). The equation governing the dynamics of this system is:

$$\frac{dx_i}{dt} = -\gamma_i x_i + F_i(X_{i1}(t), X_{i2}(t), \dots, X_{ik}(t)), \quad i = 1, \dots, N \quad (3.4)$$

where  $x_i(t)$  is a continuous variable representing the concentration of transcription factor  $i$  at time  $t$ ,  $X_i$  is a discrete binary variable ( $X_i = 1$  if  $x_i \geq \theta_i$  and  $X_i = 0$  if  $x_i < \theta_i$  where  $\theta_i$  is a threshold variable),  $\gamma_i$  is a positive decay constant and  $F_i(X_{i1}(t), X_{i2}(t), \dots, X_{ik}(t))$  is a Boolean function which depends only on  $k$  binary

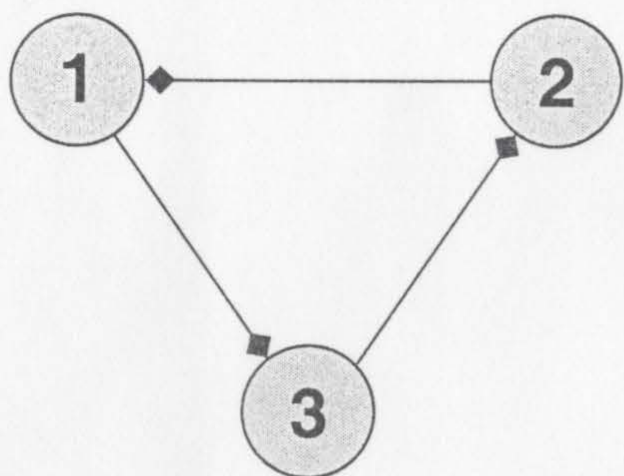


input variables. This Boolean relation between the  $k$  binary inputs and the outputted Boolean value can be summarized as a truth table as shown in Figure 3.3(b).

By setting  $\{t_1, t_2, \dots, t_k\}$  to denote the switch times when a value crosses the threshold,  $\theta_i$ , a solution to Equation 3.4 can be obtained where  $t_j < t < t_{j+1}$ :

$$x_i(t) = x_i(t_j)e^{-(t-t_j)} + F_i(X_{i1}(t), X_{i2}(t), \dots, X_{ik}(t)) \times (1 - e^{-(t-t_j)}) \quad (3.5)$$

Due to the simplicity of the dynamic equations, Glass networks are quite amenable to mathematical analysis. This class of networks can display fixed points, stable limit oscillations and chaotic dynamics (Mason et al., 2004). A detailed analysis of the dynamics and characteristics of such networks have been presented by Mestl et al. (1997), Edwards (2000), Edwards and Glass (2000), Edwards (2001), Edwards et al. (2001) and Ben-Hur and Siegelman (2004).



(a) Genetic circuit schematic of the repressilator. Nodes represent a gene / protein pair while edges represent inhibitory relationships.

Input ( $X_i$ )			Function		
1	2	3	$F_1$	$F_2$	$F_3$
0	0	0	1	1	1
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	0	0

(b) Truth table which defines the function,  $F_i$ , which implements the repressilator gene circuit.

Figure 3.3: A Boolean network model of the repressilator.

An example of a network modelled using Glass networks is the repressilator



(Elowitz and Leibler, 2000) shown in Figure 3.3(a). This circuit is an example of a synthetic gene circuit exhibiting stable oscillatory behaviour when implemented in *Escherichia coli* using plasmids. The truth table implementing the dynamics of the repressilator circuit are shown in Figure 3.3(b).

### 3.2.4 S-Systems

S-systems (synergistic and saturable system) have long been used as models of biochemical pathways, genetic networks and immune networks (Akutsu et al., 2000a). S-Systems are a class of non-linear ordinary differential equations and have the form:

$$\frac{dX_i(t)}{dt} = \alpha_i \prod_{j=1}^n X_j(t)^{g_{i,j}} - \beta_i \prod_{j=1}^n X_j(t)^{h_{i,j}} \quad (3.6)$$

where  $\alpha$  and  $\beta$  are rate constants and  $g$  and  $h$  are exponential parameters referred to as kinetic orders. S-Systems have unique mathematical properties allowing large realistic phenomena to be investigated and can be derived from general mass balance equations by aggregating inputs and outputs approximated by the products of power-law functions. Each dimension of the S-System model represents the dynamics of a single variable represented as the difference of two products of power-law functions – one describing the influxes and the other describing the effluxes. This can also be thought of as a linearization of the logarithmic space (exploited by Akutsu et al. (2000a)). Only terms that directly influence a particular influx or efflux are included in the model. However, the general structure of the model always remains the same which has led to the development of numerous analytical tools for simulating, deriving and analyzing such models.

Although S-Systems purport to more accurately model the biological processes



inherent in regulatory networks, they still cannot handle some important concepts such as complex enzymatic reactions (but neither do the other methods presented). Another weakness with the S-System formalism is the number of parameters required. In general, for an  $n$ -dimensional vector,  $n \times (2n+2)$  parameters are required in order to specify the dynamics of the system. In addition, since the system is non-linear, many traditional optimization schemes are excluded or must operate on linearized versions of the problem at or near equilibrium points to obtain reasonable approximations of the solution (such as that presented in the next section).

### 3.2.5 Control Theory / State Space Models

Modern control theory or the state-space method of description for ODEs has been in existence for over 100 years. This method can be summarized as follows:

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)\end{aligned}\tag{3.7}$$

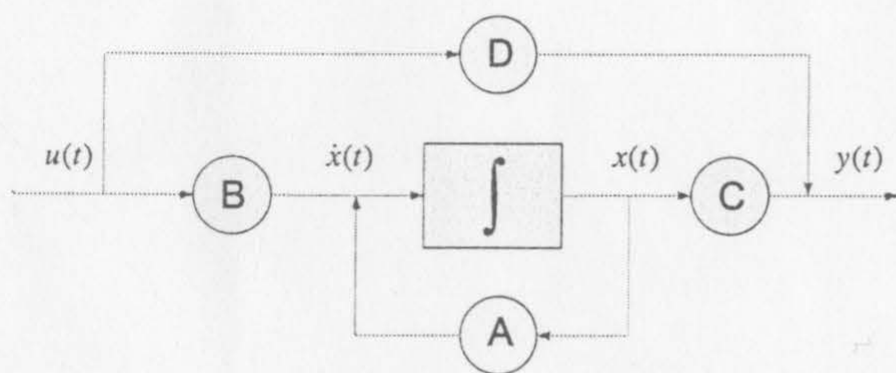


Figure 3.4: The relation between the matrices  $\mathbf{A}$ ,  $\mathbf{B}$ ,  $\mathbf{C}$ ,  $\mathbf{D}$ , the state vector,  $\mathbf{x}(t)$ , the output vector,  $\mathbf{y}(t)$  and the input vector,  $\mathbf{u}(t)$  in control theory.

Equation 3.7 relates the state vector  $\mathbf{x}(t)$  with its time derivative and control input  $\mathbf{u}(t)$ . The matrices  $\mathbf{A}$  and  $\mathbf{C}$  are the dynamics matrices, while matrices  $\mathbf{B}$  and  $\mathbf{D}$  are the input matrices. The state vector, along with initial conditions completely



characterizes the current state of the system as well as all future states. Therefore, the future states of the system are dependent only on the current state and future inputs to the system. The state vector can thus be viewed as representing the current state of the system with the values of the state vector at different time points representing a trajectory in the  $n$ -dimensional vector space (thus the name “state-space” method). The first equation defines the internal dynamics of the system while the second equation defines the dynamics visible to the observer (which may be different from the first equation). This is shown in schematic form in Figure 3.4.

In using such an approach, the states often correspond to the “concentrations” of genes and / or proteins. However, due to the large number of genes in such a model, it is common in such approaches to take states to mean either groups of genes or to represent certain genetic factors in the system. This is often done in order to reduce the complexity and computational requirements of such models and is typically accomplished using clustering or projection methods which are briefly reviewed in Appendix B. State-space methods have been used in the reverse-engineering of regulatory networks by Rangel et al. (2001, 2004a,b) and Wu et al. (2004).

### 3.3 Stochastic Models

Stochastic models of gene regulatory processes purport to remedy many of the shortcomings of deterministic (mainly differential equation) based approaches. One such shortcoming is the assumption of a continuous rate of protein production. Typically, there are a small number of transcription factors in a system which is not well represented by continuous models. In fact, proteins are not produced at a continuous rate at all but rather in short bursts (McAdams and Arkin, 1997). In addition, the thresholds at which gene activation / inhibition occur can be crossed at different times even



in the same populations of genes. Some mechanisms of transcriptional regulation are known to amplify noise creating heterogeneity within a population.

With the addition of noise in gene transcription, individual cells may take different regulatory paths despite having the same regulatory input (Elowitz et al., 2002). Therefore, it is likely that evolution has selected networks which can produce deterministic behaviours from stochastic inputs in a noisy environment. In fact, certain topologies in networks can attenuate the effects of noise (such as the feedback loop) (Rao et al., 2002) and also that noise can indeed act as a stabilizer itself in other systems (Hasty et al., 2000).

There are generally two methods for modelling stochastic gene regulation. The first are stochastic differential equations:

$$\frac{dx_i}{dt} = f_i(x_i) + \nu_i(t) \quad (3.8)$$

Equation 3.8 gives the form of a stochastic differential equation that explicitly models noise in the system through the term  $\nu(t)$ . Equation 3.8 is referred to as the Langevin equation and is generally not amenable to solution by either analytical or numerical means. Typically, solutions to the Langevin equations are obtained through the use of Monte-Carlo algorithms.

The second approach is to characterize the transitions of a molecule using probability functions. During each individual time step, a molecule is given a certain probability of transitioning to a different discrete state. From this, a probability density function for the behaviour of the system can be obtained. Such systems are referred to as the “Master Equation” and are often solved by techniques such as the Gillespie algorithm (McAdams and Arkin, 1998).

Although stochastic models are often more realistic than their deterministic coun-



terparts, they are expensive to simulate. In fact, for many realistically sized systems, stochastic approaches are impractical (Swain, 2005). However, stochastic models of gene regulation have been successfully used in Keasling et al. (1995), McAdams and Arkin (1998) and Kastner et al. (2002) to show but a few examples. A good review of stochastic genetic networks can be found in (McAdams and Arkin, 1997, 1998, Kepler and Elston, 2001).

### 3.4 Artificial Regulatory Network Model

In this section, a regulatory network model referred to as the artificial regulatory network (ARN) model first presented by Banzhaf (2003a,b) is introduced. The ARN consists of a bit string representing a genome with direction (i.e.  $5' \rightarrow 3'$  in DNA) and mobile “proteins” which interact with the genome through their constituent bit patterns. In this model, proteins are able to interact with the genome, most notably at “regulatory” sites located upstream from genes. Attachment to these sites produces either inhibition or activation of the corresponding protein. These interactions may be interpreted as a regulatory network with proteins acting as transcription factors.

A “promoter” bit sequence of 8-bits was arbitrarily selected to be “01010101”. By randomly choosing “0”s and “1”s to generate a genome, any one-byte pattern can be expected to appear with probability  $2^{-8} = 0.39\%$ . Since the promoter pattern itself is repetitive, overlapping promoters or periodic extensions of the pattern are not allowed, i.e. a bit sequence of “0101010101” (10-bits) is detected as a single promoter site starting at the first bit. However regions associated with one gene may overlap with another should a promoter pattern also exist within a portion of the coding region of a gene.

The promoter signals the beginning of a gene on the bit string analogous to an



open reading frame (ORF) on DNA – a long sequence of DNA that contains no “stop” codon and therefore encodes all or part of a protein. Each gene is set to a fixed length of  $l_{gene} = 5$  32-bit integers which results in an expressed bit pattern of 160-bits.

Immediately upstream from the promoter sites exist two additional 32-bit segments which represent the enhancer and inhibitor sites. As previously mentioned, attachment of proteins (transcription factors) to these sites results in changes to protein production for the corresponding genes (regulation). In this model, it is assumed that there exists only one regulatory site for the increase of expression and one site for the decrease of expression of a given protein. This is a radical simplification since natural genomes may have five to ten regulatory sites per gene that may even be occupied by complexes of proteins (Banzhaf, 2003a).

Processes such as transcription, diffusion, spatial variations and elements such as introns, RNA-like mobile elements and translation procedures resulting in a different alphabet for proteins are neglected in this model. This last mechanism is replaced as follows: Each protein is a 32-bit sequence constructed by a many-to-one mapping of its corresponding gene which contains five 32-bit integers. The protein sequence is created by performing the majority rule on each bit position of these five integers so as to arrive at a 32-bit protein. Ties (not possible with an odd number for  $l_g$ ) for a given bit position are resolved by chance.

Proteins may then be examined to see how they may “match” with the genome, specifically at the regulatory sites. This comparison is implemented using the XOR operation which returns a “1” if bits on both patterns are complementary. The degree of match between the genome and the protein bit patterns is specified by the number of bits set to “1” during an XOR operation. In general, it can be expected that a Gaussian distribution results from measuring the match between proteins and bit sequences in a randomly generated genome (Banzhaf, 2003a). By making the simpli-



fying assumption that the occupation of both of a gene's regulatory sites modulates the expression of its corresponding protein, a gene-protein interaction network may be deduced comprising the different genes and proteins which can be parameterized by strength of match. The bit-string for one gene is shown in Figure 3.5.

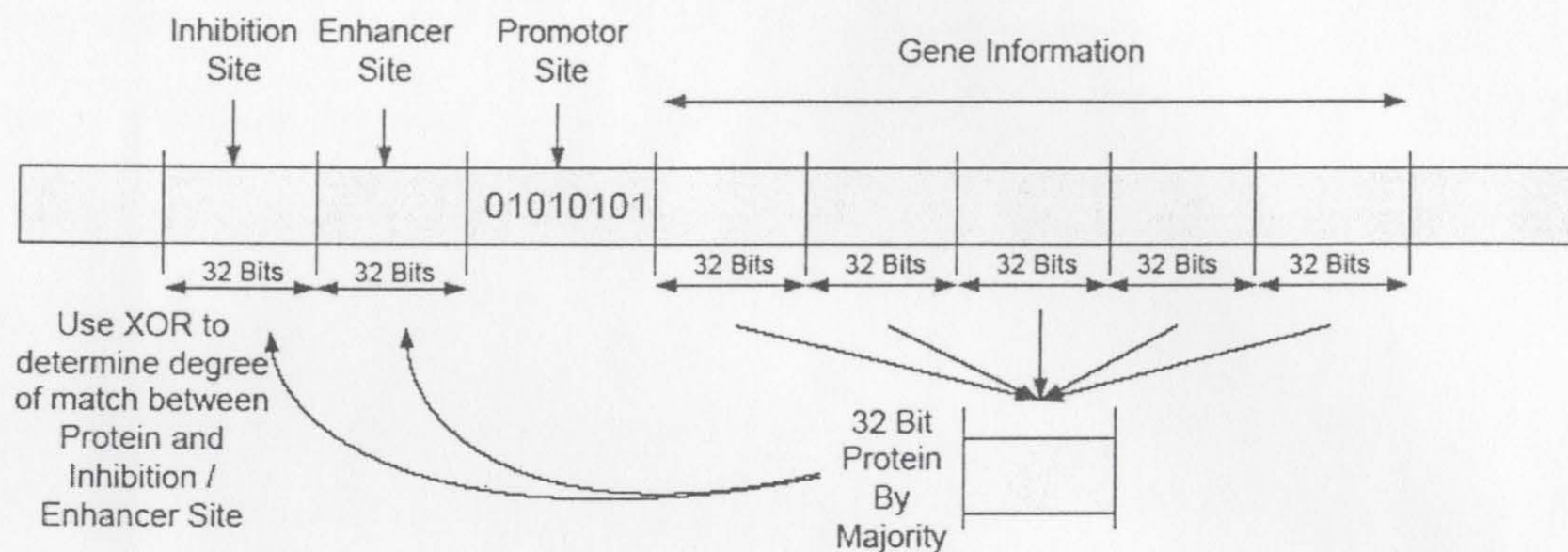


Figure 3.5:

The rate at which protein  $i$  is produced is given by:

$$\frac{dc_i}{dt} = \frac{\delta (e_i - h_i) c_i}{\sum_j c_j} \quad (3.9)$$

$$e_i, h_i = \frac{1}{N} \sum_j^N c_j \exp(\beta(u_j - u_{max})) \quad (3.10)$$

where  $e_i$  and  $h_i$  represent the excitation and inhibition of the production of protein  $i$ ,  $u_j$  represents the number of matching bits between protein  $j$  and activation or inhibition site  $i$ ,  $u_{max}$  represents the maximum match (in this case, 32),  $\beta$  and  $\delta$  are positive scaling factors, and  $c_i$  is the concentration of protein  $i$  at time  $t$ . The concentrations of the various proteins are required to sum to 1. This ensures that there is a competition between binding sites for proteins.

The ARN model also bears some resemblance to a recurrent neural network



(RNN). The ARN genes and the match strength between TF binding sites and TFs are analogous to the neurons and connection weights in an RNN.

### 3.5 Conclusion

As previously noted, the choice of a formalism for modelling genetic regulatory networks is problem domain dependent. Although Boolean network models have been used in modelling gene regulation, there is some debate as to their utility due to many of the assumptions inherent in the model. Such assumptions include synchronous updating, all-or-nothing gene transcription, and restrictions on network connectivity.

Differential equation models eliminate many of the limitations of Boolean models but add increasing levels of complexity necessitating the selection of parameter values and the use of complex analysis techniques.

In turn, stochastic models purport to more accurately model genetic regulation by removing the deterministic assumptions inherent in differential equation models albeit with an additional penalty of computational complexity and analysis. However, Kim and Tidor (2003) have shown that behaviours found in some systems cannot be reconciled with common models of genetic regulation. Therefore, care must be taken when interpreting the results of a study using any formalism.

Although the most common formalisms were reviewed in this chapter, there exist several formalisms in current use for modelling genetic regulatory networks that were not discussed. Some examples are the use of petri nets, and neural networks.



## Chapter 4

# Topological Characterization

Since genetic regulatory networks are typically characterized by a large number of interactions (usually excitatory or inhibitory) between regulatory elements, studying the topology of such networks may prove to be informative. Such an approach has been pursued by Wuchty et al. (2003), Guelzim et al. (2002), Milo et al. (2002), Shen-Or et al. (2002), Babu and Teichmann (2003), Bray (2003), Mangan and Alon (2003), Wolf and Arkin (2003), Yu et al. (2003), Berg and Lassig (2004), Dobrin et al. (2004), Hahn et al. (2004), Kashtan et al. (2004a), Kuo and Banzhaf (2004), Babu et al. (2004), Milo et al. (2004), van Noort et al. (2004), Vazquez et al. (2004), Yeger-Lotem et al. (2004) and Yu et al. (2004).

Since one of the most basic features of any complex network is its structure, it is natural to investigate its connectivity. The structure of networks are often constrained and shaped by the growth processes that create them (including evolution in the case of natural networks). Studying the topology of such networks might shed some light on the possible structures and dynamics which have been exploited by nature.

Of course, in order to study the topology of such networks an abstraction must be made such that a GRN can be represented by a series of nodes and edges. Some



of these abstractions were discussed in Chapter 3. Typically, nodes in such an abstraction represent individual genes (and their associated proteins) while the directed edges which connect the nodes represent one gene's effect (excitatory or inhibitory) on another as shown in Figure 3.3(a).

## 4.1 Scale-Free Network Topologies

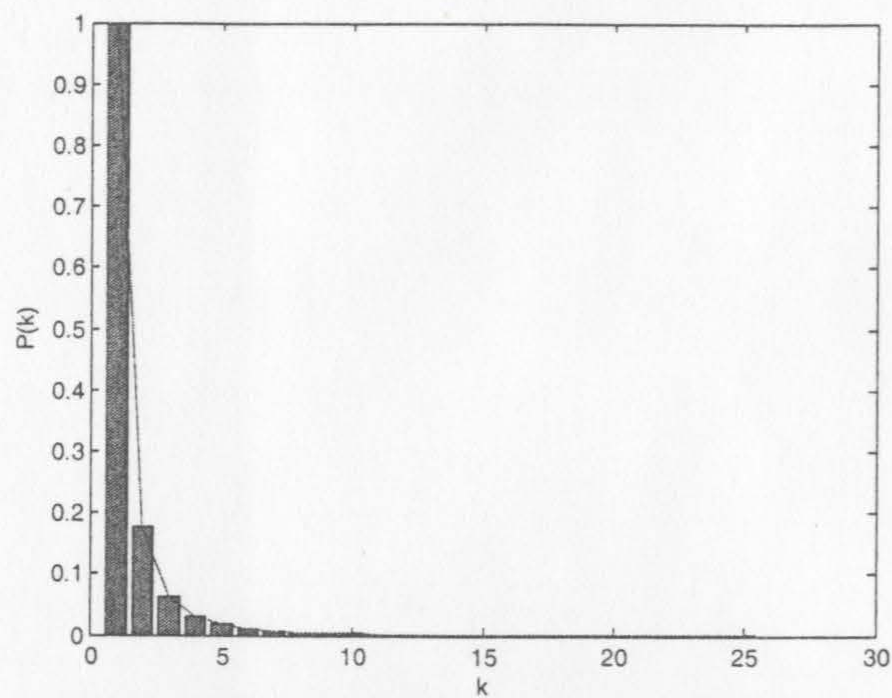
A high degree of self-organization may characterize the large-scale properties of complex networks (Barabási and Albert, 1999). Many researchers have shown that the probability,  $P(k)$  (the number of nodes connected to  $k$  other nodes in a network), decays as a power-law, following  $P(k) \sim k^{-\gamma}$  where  $\gamma$  is a constant.

This has been shown in systems as diverse as the internet (Faloutsos et al., 1999), protein interaction networks (Wuchty, 2001), the electrical power grid of the western United States of America (Watts, 2003), the neuronal network of the worm *Caenorhabditis elegans* (Watts, 2003), the network of citations of scientific papers (Barabási et al., 2002), metabolic networks (Jeong et al., 2000), the *Saccharomyces cerevisiae* co-expression network (Guelzim et al., 2002, van Noort et al., 2004) and transcriptional regulatory networks (Bray, 2003, Babu et al., 2004). This is in contrast to random networks (so-called Erdős-Rényi graphs) which follow a Poisson degree distribution,  $p(k) \sim \frac{\lambda^k \exp(-\lambda)}{k!}$ .

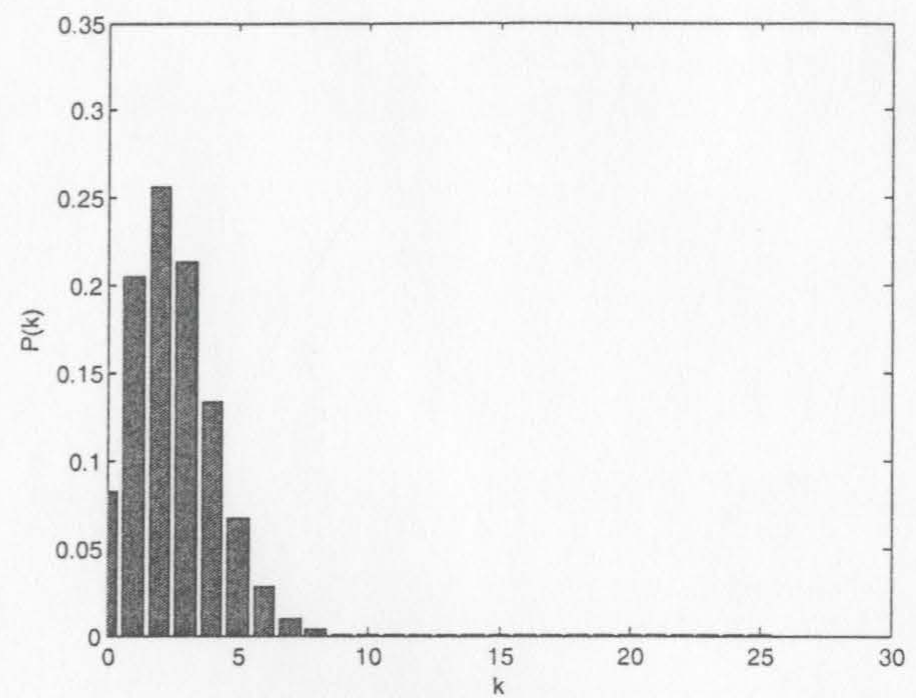
Figure 4.1(a) shows an example of a degree distribution which follows a power-law distribution while Figure 4.1(b) shows one which follows a Poisson distribution. An example of the graph topology of a scale-free network is shown in Figure 4.2.

It has been suggested that scale-free networks emerge in the context of a dynamic network with the addition of new vertices connecting preferentially to vertices which are highly connected in the network (Barabási and Albert, 1999), as well as





(a) Power-law distribution where  $\gamma = 2.5$ ,  $k$  is the vertex degree, and  $P(k)$  is the probability of a vertex having degree  $k$ .



(b) Poisson distribution where  $\lambda = 2.5$ ,  $k$  is the vertex degree, and  $P(k)$  is the probability of a vertex having degree  $k$ .

Figure 4.1: Examples of two different degree distributions.

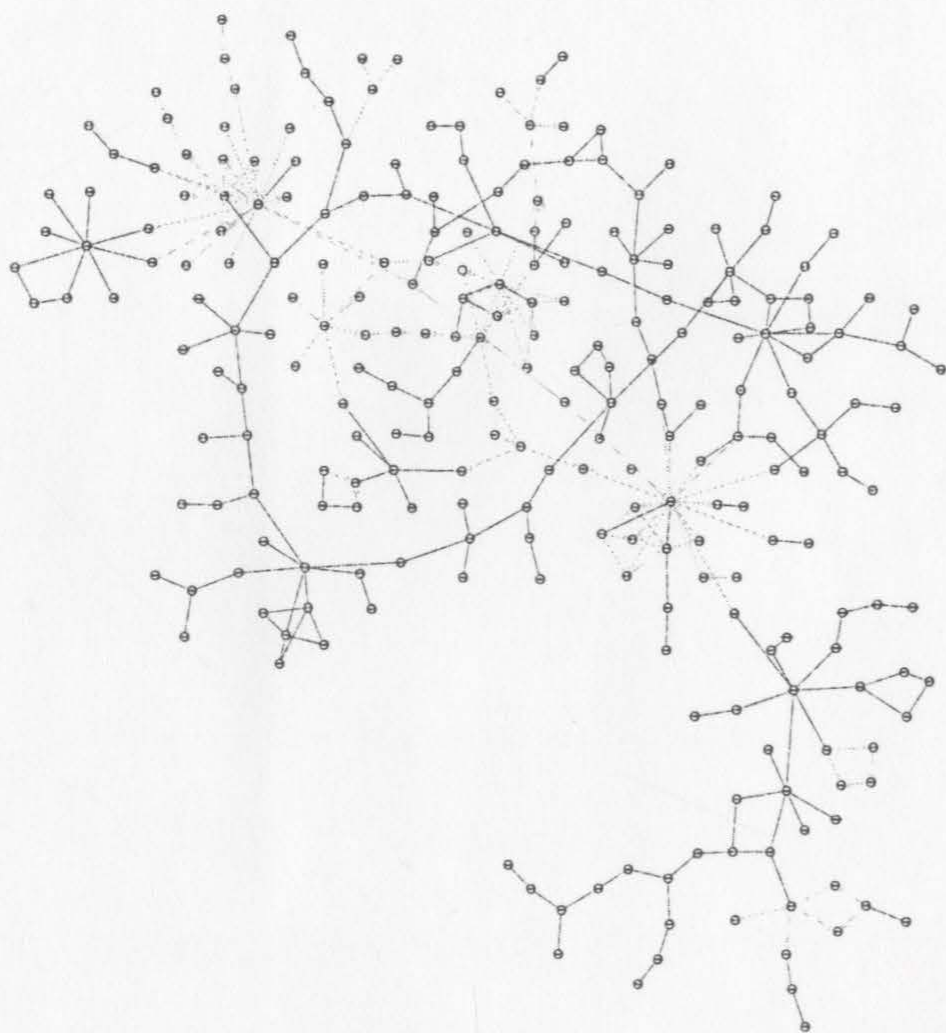


Figure 4.2: An example of a network displaying scale-free characteristics. The graph is of the protein-protein and protein-DNA interactions among 332 yeast genes (Shannon et al., 2003).



through explicit optimization (Valverde et al., 2002) and duplication and divergence (Romualdo et al., 2003, Kuo and Banzhaf, 2004).

One possible explanation for the apparent ubiquity of scale-free topologies is that such networks should be more robust to random failure of individual nodes. Since the vast majority of nodes in the network are connected to few other nodes, a failure among these nodes would be unlikely to dramatically affect the functioning of the network at a global scale. However, this also means that the few highly connected nodes or hubs are particularly vulnerable to targeted attack since their failure would have drastic effects on network behaviour. In fact, the properties of scale-free networks and their robustness to individual node failure has been studied in the internet by Albert et al. (2000) and in protein networks by Solé et al. (2002) and Jeong et al. (2001).

However, a study by Yu et al. (2004) found that the so-called hubs in genetic transcriptional regulatory networks were not essential to organism survival (although the opposite was found in this paper for proteins and in Jeong et al. (2001)). In fact, in a study of the genetic regulatory network of *Escherichia coli* by Hahn et al. (2004), no correlation could be found between evolutionary rate and highly connected proteins. In addition, only a weak correlation exists (a correlation could be found only for genes involved in the cell cycle and transcription) for the protein interaction network of *Saccharomyces cerevisiae* (highly connected proteins could tolerate as many amino acid substitutions as any other protein). These results suggest that power-law distributions in cellular networks do not reflect selection for mutational robustness. Such findings prove a valuable cautionary note for graph theorists since these results directly contradict theories on the evolutionary benefits of scale-free topologies which were proved analytically and in simulation.



## 4.2 Small-World Network Topologies

Watts (2003) defines small-world graphs as any graph with  $n$  vertices and average vertex degree  $k$  that exhibits  $L \approx L_{\text{random}}(n, k) \sim \frac{\ln(n)}{\ln(k)}$  and  $C \gg C_{\text{random}} \sim \frac{k}{n}$  for  $n \gg k \gg \ln(n) \gg 1$ .  $C$  is the clustering coefficient which is defined as follows:

$$\text{if vertex } v \text{ has } k_v \text{ neighbours, } C = \frac{2}{n} \sum_{v=1}^n \left( \frac{k_v(k_v - 1)}{2} \right) \quad (4.1)$$

$L$  is the characteristic path-length of the network (average number of links connecting two nodes).  $L_{\text{random}}$  and  $C_{\text{random}}$  refer to the characteristic path-length and clustering coefficient for a random graph with the same  $k$  and  $n$  respectively.

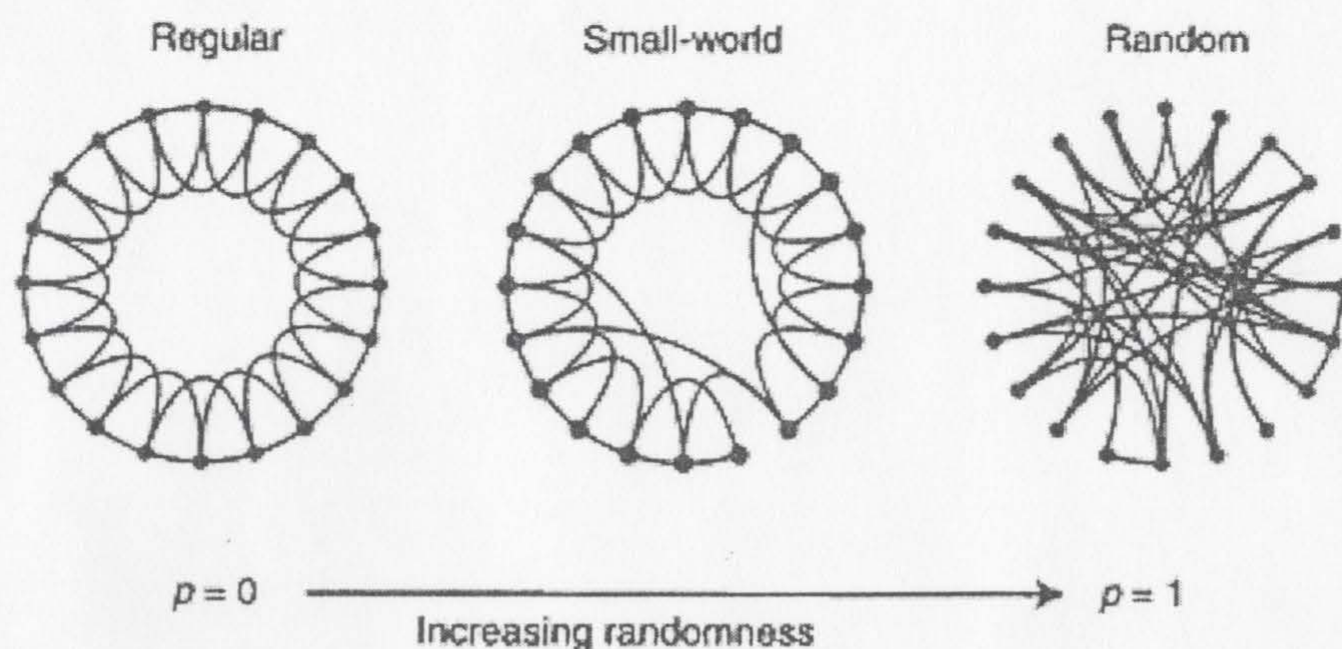


Figure 4.3: Networks generated from a regular lattice (left-most graph). As random rewiring occurs at a rate of  $p$ , a small-world network is obtained for values of  $p$  intermediate between 0 and 1. Reprinted with permission of Watts (2003).

Like scale-free network topologies, small-world topologies have also been noted in many networks (including those with scale-free topology) such as the electrical power grid of the western United States of America (Watts, 2003), the neuronal network of



the worm *Caenorhabditis elegans* (Watts, 2003), the network of film actors who have acted in the same films (Watts, 2003), and the *Saccharomyces cerevisiae* co-expression network (van Noort et al., 2004).

Watts (2003) has also shown that models of dynamical systems which display small-world coupling show enhanced signal-propagation speed, computational power, and increased synchronizability. In addition, Watts (2003) has shown that infectious diseases spread more rapidly and easily in small-world networks than in regular lattices or random topologies and may be reasonable models of such processes.

Thus, it has been suggested that small-world network topologies are also prevalent in natural systems and that the study of such networks might increase our understanding of the ways in which natural systems function (Watts and Strogatz, 1998).

### 4.3 Network Motifs

The previous two topological measures characterize network topology at the global level. In contrast, local graph properties have also been proposed for studying networks. There has been significant interest in studying static network motifs as a tool for understanding regulatory networks (Mangan and Alon, 2003, Milo et al., 2002, Shen-Or et al., 2002, Wuchty et al., 2003, Wolf and Arkin, 2003, Yu et al., 2003, Banzhaf and Kuo, 2004, Berg and Lassig, 2004, Dobrin et al., 2004, Kashtan et al., 2004a, Milo et al., 2004, Vazquez et al., 2004, Yeager-Lotem et al., 2004).

Network motifs are usually defined as the structural elements (subgraphs) which occur in statistically significant quantities as compared to random networks (Milo et al., 2002). This is in contrast to sequence motifs which identify common sequences in genes at the DNA level. The possible implications of having certain subgraphs being found in greater abundance than would be expected in similar random networks is



that these local network motifs may convey some sort of functional advantage to the system. Such subgraphs form the basic elements of more complex networks.

Whereas one or more edges connect two nodes of a graph (where the nodes represent gene / protein pairs and the edges interactions between them), network motif analysis typically starts with three nodes (or more) and their corresponding connections. It has been proposed that studying “network motifs” can lead toward a better understanding of the potential basic structural elements which make up complex networks. In fact, several motifs such as the bi-fan (Kashtan et al., 2004b), the feed-forward loop (Mangan and Alon, 2003) and the feedback loop (Kashtan et al., 2004b) have been the subject of study.

If we are to believe the contention that network motifs confer some form of functional advantage to an organism, we might expect such connectivity to be preserved over evolutionary time. There is some conflicting evidence regarding whether this is actually the case. In the protein interaction network of *Saccharomyces cerevisiae*, Wuchty et al. (2003) concluded that motifs could be evolutionary conserved topological units of cellular networks. However, Babu et al. (2004) found that network motifs in transcriptional regulatory networks were not preferentially conserved over evolutionary time for a variety of different organisms.

In addition, it has been implied that the distribution of subgraphs for given network domains (such as transcriptional regulatory networks, social networks) are distinct enough to allow classification (Banzhaf and Kuo, 2004, Milo et al., 2004). This implies that certain network motifs are more prevalent in a given type of network (presumably since their presence is somehow beneficial at the global level).

Appendix E lists all three-node connection patterns in directed graphs, including auto-regulatory connections, up to isomorphism. Appendix G lists some four-node connection patterns in directed graphs, including auto-regulatory connections, up to



isomorphism. A presentation of a full table of all four-node connection patterns is impractical due to space limitations.

## 4.4 Conclusion

There has been a large body of work regarding the characterization of network topologies in general as well as specifically relating to regulatory networks. In fact, Vazquez et al. (2004) has suggested that both local and global network properties are predictive of each other. Specifically, that network motifs can be predictive of large scale topology (scale-free and small-world topology) and vice versa.

The evidence of Babu et al. (2004) would seem to indicate that transcriptional regulatory networks evolve in a step-like manner where the gain or loss of individual connections plays a greater role than a similar gain or loss in whole motifs or submodules. However, as is the case for scale-free networks and mutational robustness, it may be that the supposed benefits of such topological conformations only have beneficial effects at the protein interaction level as opposed to the level of transcriptional regulation. In this regard, more work needs to be done to elucidate the relationship between the topology and function of transcriptional regulatory networks.

In addition, the network measures discussed in this chapter have been on static network topologies. However, transcriptional regulation is a dynamic process with some genes taking part in some mechanisms and not in others (including during the normal cell cycle). Luscombe et al. (2004) found that the vast majority of hubs in the transcriptional regulatory network of *Saccharomyces cerevisiae* acted transiently and only for certain cellular conditions (these hubs only acted as hubs for certain processes) over a wide range of conditions. This would seem to indicate that more work needs to be done on characterizing the topology of dynamical networks.



## Chapter 5

# Network Topologies in the ARN Model

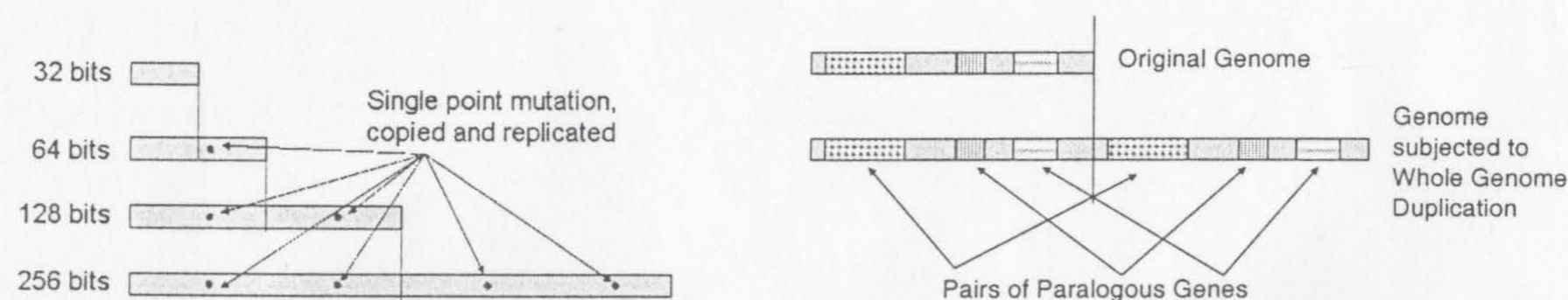
As was previously stated, the choice of formalism when modelling regulatory networks largely depends on the aims of the study. This chapter investigates how duplication and divergence affect the topology of a regulatory network. As such, the ARN model of Chapter 3 is the most appropriate choice. With this model, duplication and divergence can be more directly implemented on the genetic string as opposed to directly on the network (gene duplication happens directly on the genome level in nature). In addition, topological relationships can be easily investigated by the parameterization of the threshold. The presence of scale-free, small-world and network motif topologies is then investigated in the ARN model. Portions of the work presented in this chapter have been previously published in Banzhaf and Kuo (2004) and Kuo and Banzhaf (2004).



## 5.1 Gene Duplication and the ARN Model

The mechanism of gene duplication has been recognized as being important in supplying raw material for biological evolution since the 1930's (Zhang, 2004) or even 1910 according to Taylor and Raes (2004). However, it was the seminal work of Ohno (1970) which popularized this idea among biologists. A comprehensive history of research in gene duplication before the work of Ohno (1970) can be found in Taylor and Raes (2004).

In the case of the ARN model, the genome is created through a series of whole length duplication and divergence events. First, a random 32-bit string is generated. This string is then used in a series of length duplications similar to those found in natural systems (Nadeau and Sankoff, 1997, Wolfe and Shields, 1997, Taylor and Raes, 2004) followed by mutations in order to generate a genome of length  $L_G$ . An illustration of this process is given in Figure 5.1(a).



(a) Effect of duplication on a single mutation in the genome.

(b) Whole genome duplication creates pairs of functionally redundant paralogous genes. One gene of each paralogous pair is now free to diverge either disappearing or acquiring a new function without affecting the original function of the gene.

Figure 5.1: Whole genome duplication and divergence.

The duplication and divergence mechanism used in the ARN model is most similar to so-called whole genome duplication. Ohno (1970) suggested that whole genome



duplication might be an important evolutionary mechanism for generating novelty in the genome and additionally might give a reasonable explanation for speciation. When whole genome duplication occurs, pairs of functionally redundant paralogous genes are created. This is shown in Figure 5.1(b).

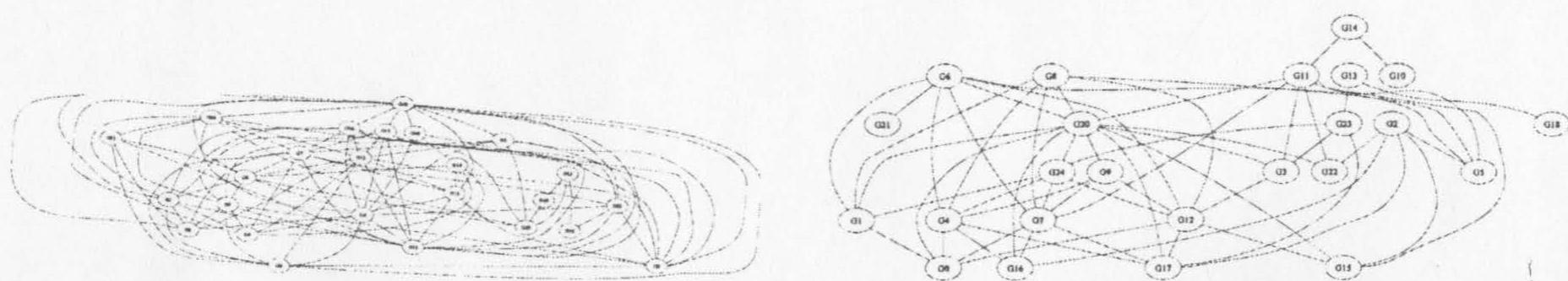
Since only one pair of paralogous genes is required to retain its original function, the second is free to diverge. This might lead to the second gene being lost or acquiring a novel function through subsequent mutations. A review of the role of gene duplication in the creation of novel proteins can be found in Hughes (2005). In fact, evidence for either whole genome duplications or substantial gene duplication events exist in the literature. Specifically, there has been evidence for gene duplications in *Saccharomyces cerevisiae* (Wolfe and Shields, 1997, Friedman and Hughes, 2001, Gu et al., 2002, Dujon et al., 2004, Kellis et al., 2004, Teichmann and Babu, 2004) (and in simulation by van Noort et al. (2004)), *Escherichia coli* (Friedman and Hughes, 2001, Babu and Teichmann, 2003, Babu et al., 2004, Teichmann and Babu, 2004), vertebrates (Nadeau and Sankoff, 1997) and other organisms. More generally, three quarters of the transcription factors in *Escherichia coli* have arisen from gene duplication (Babu and Teichmann, 2003) and at least 50% of prokaryotic genes and over 90% of eukaryotic genes are created by gene duplication (Teichmann and Babu, 2004). A review of the mechanisms that may facilitate gene duplications can be found in Zhang (2004) and are beyond the scope of this thesis.

In addition, some of the properties of gene duplication are investigated in a mathematical framework by Wagner (1994). It was found through this analysis that the evolution of gene networks should occur preferentially by either the duplication of single genes or by duplication of all genes involved in the network. Romualdo et al. (2003) and Chung et al. (2003) also both found that duplication models could account for scale-free connectivity seen in biological networks.



## 5.2 Scale-Free & Small-World Topologies in the ARN Model

As was noted in Chapter 3, the effect of one gene's products on another can be investigated in the ARN model by looking at the degree of match between one gene's protein and another's regulatory sites (one excitatory and one inhibitory site). By examining the interaction networks of the ARN model (created through the whole genome duplication and divergence mechanism) at different matching strengths (thresholds), different network topologies are obtained. An example is shown in Figures 5.2(a) and 5.2(b). Each node in the diagram represents a gene found in the genome along with its corresponding protein forming a gene-protein pair. Edges in the diagram represent a regulatory influence of one gene's protein on another gene. For the diagrams presented, a genome was created by the previously mentioned duplication and divergence procedure with the network interaction diagrams being created at thresholds of 21 and 22.



(a) Gene-protein interaction network for a random genome at a threshold of 21 bits.

(b) Gene-protein interaction network for a random genome at a threshold of 22 bits.

Figure 5.2: Gene-protein interaction networks generated by two different thresholds.

Although the actual genome has not changed, by simply changing the threshold parameter, different network topologies are obtained. The reader may notice that the diagrams in Figures 5.2(a) and 5.2(b) possess different numbers of genes and proteins. This is due to the fact that only connected gene-protein pairs are displayed in the



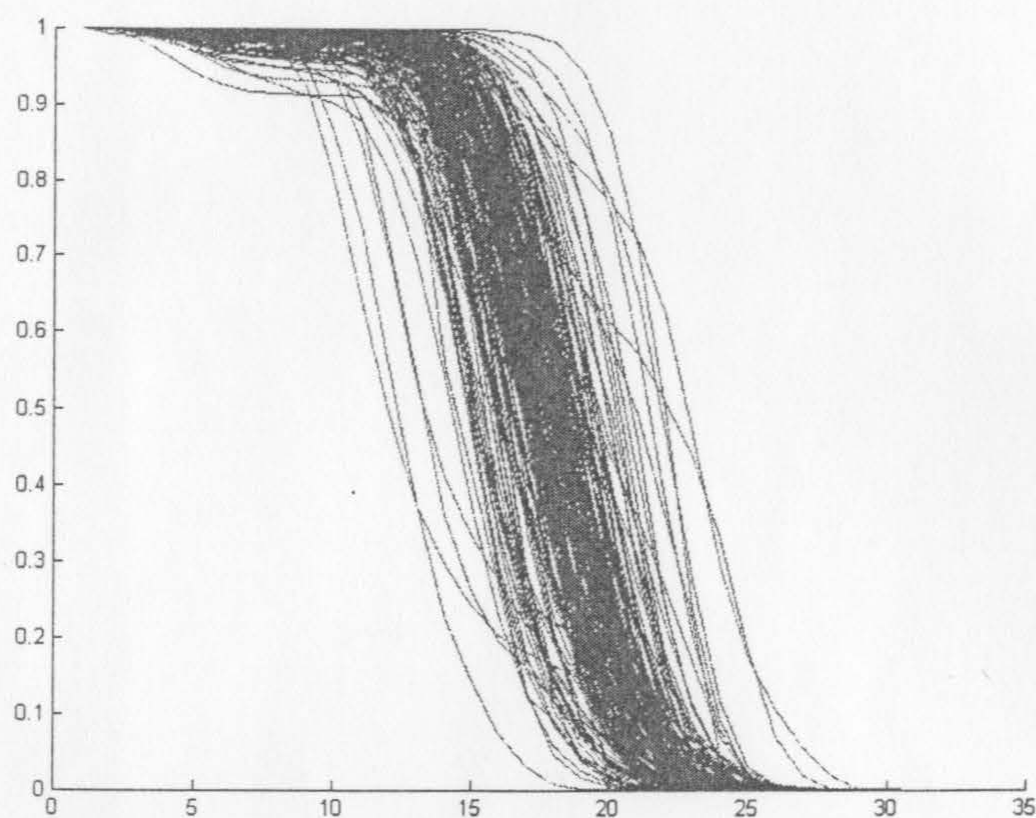


Figure 5.3: Diagram showing the fraction of edges in a graph at a given threshold ( $x$ -axis) compared to a fully connected graph for 200 networks generated by duplication and divergence.

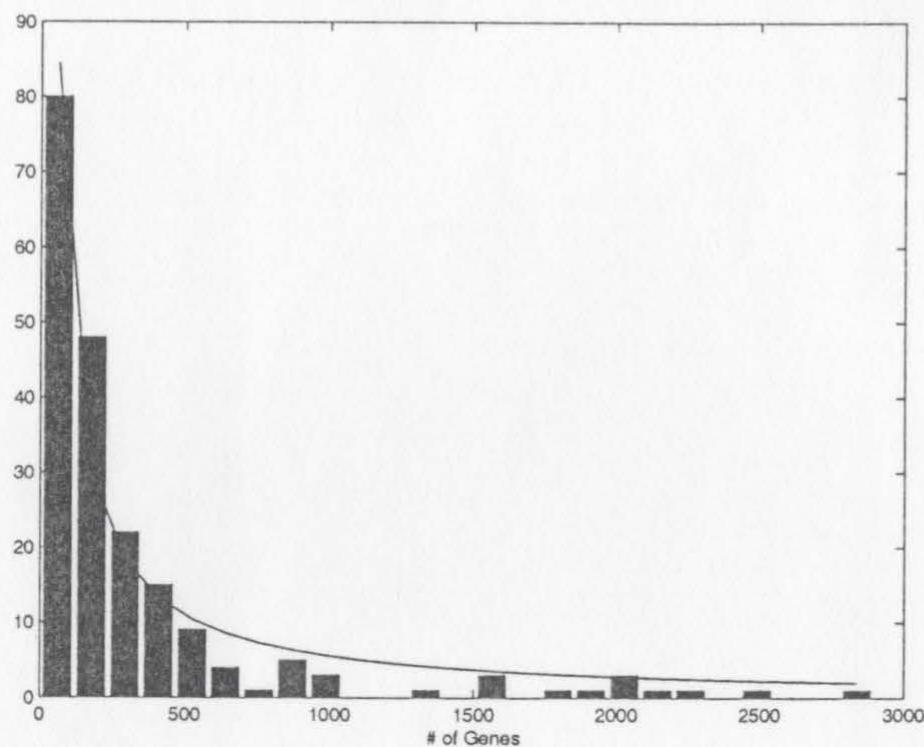
diagrams. Should a change in the parameterized threshold lead to the creation of an isolated node, it is deleted from the diagram. Only the largest network of interactions is displayed.

It is possible to have multiple clusters of gene-protein interactions that are not interconnected. This is likely to occur as the threshold level is increased. As connections between gene-protein pairs are lost due to the threshold, each cluster of gene-protein pairs begins to become isolated from the others. This often occurs abruptly indicating a phase transition between sparse and full network connectivity. The relationship between the number of edges in the graph and the threshold are shown in Figure 5.3. As the threshold increases from 0 to 32 (the  $x$ -axis), the fraction of edges in the graph over the number of edges in a fully connected network of the same number of nodes (also the number of edges in any ARN graph at threshold 0) goes from 1.0 to 0.0. We can observe a sharp transition from full connectivity to no connectivity.

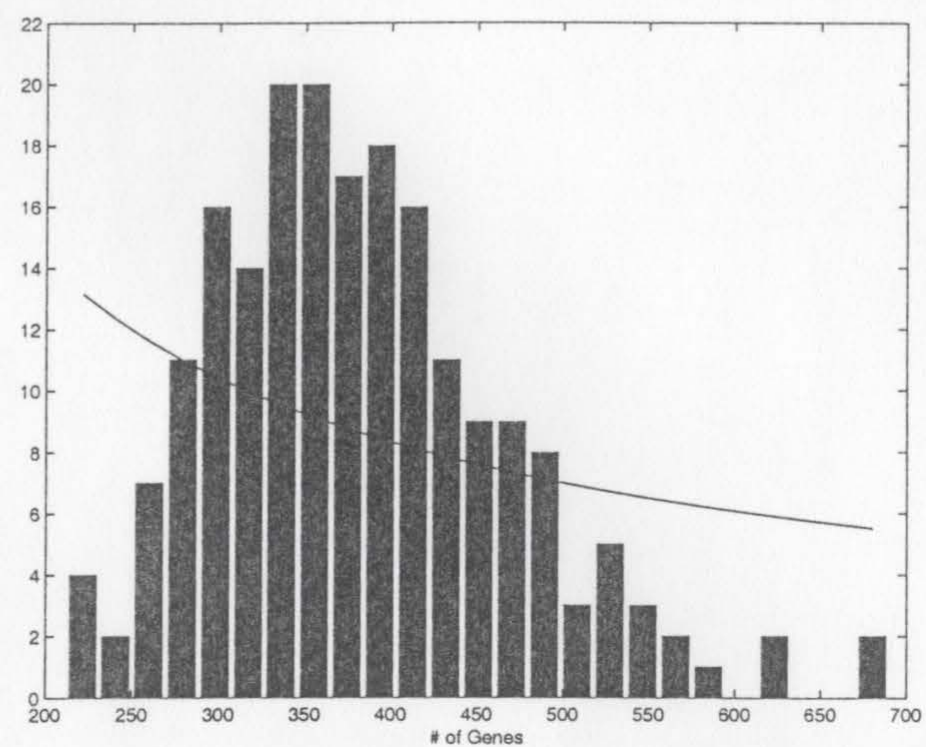
In order to generate such networks, a divergence (or mutation) rate for the duplication and divergence mechanism must be chosen. First, mutation rates of 1% and 5% were examined. 200 genomes were generated by 12 duplication events per



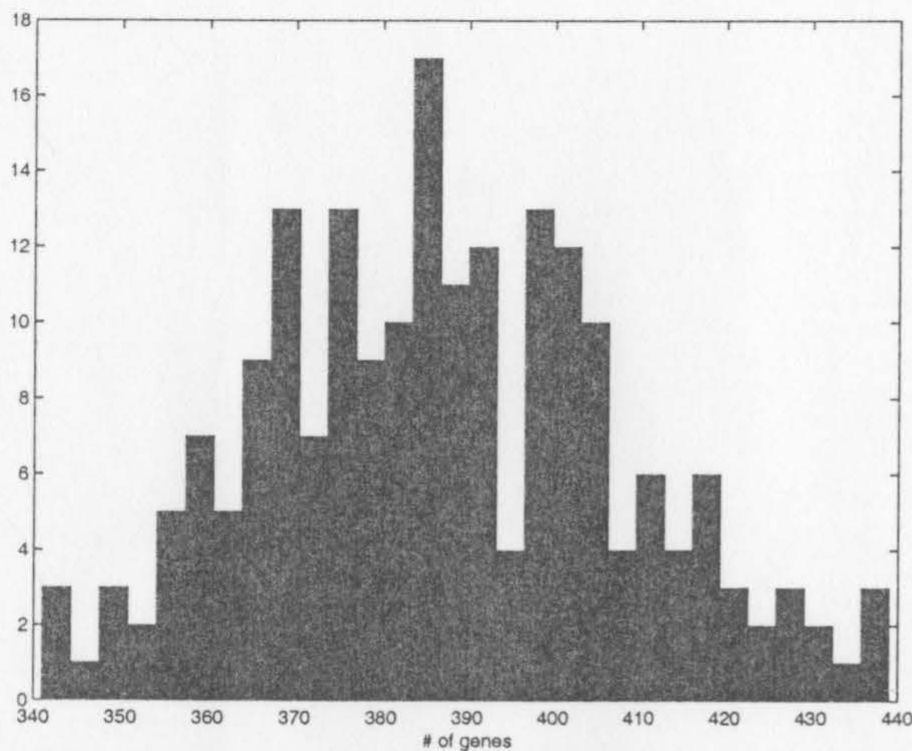
genome leading to individual genomes of length  $L_G = 131072$ . From these genomes, the number of genes were then determined based on the number of promoter patterns present. The distribution of the number of genes present in the genome of size  $L_G$  is shown in Figures 5.4(a) and 5.4(b).



(a) Histogram of the number of genes in each genome (200 genomes) fitted to a power-law:  $P(g) \sim g^{-\gamma}$  for a mutation rate of 1.0%.  $\gamma$  was calculated to be 0.9779.



(b) Histogram of the number of genes in each genome (200 genomes) fitted to a power-law:  $P(g) \sim g^{-\gamma}$  for a mutation rate of 5.0%.



(c) Histogram of the number of genes in 200 genomes whose bits have been chosen at random.

Figure 5.4: Histogram of the number of genes in the ARN model.

It can be observed that the distribution of the number of genes in Figure 5.4(a)



follows a power-law-like distribution. However, in Figure 5.4(b) the apparent distribution is disrupted. This is attributed to the higher rate of mutation. At such a mutation rate, the rewiring of the network becomes so prevalent that it begins to disrupt the duplication of nodes leading to a randomly connected network.

For an 8-bit promoter, the probability that it remains intact after one duplication event is only 66% at a mutation rate of 5%. Therefore, it can be expected that many of the genes copied during the duplication process will be subsequently destroyed in later duplication steps. However, there will also be other genes which arise from this higher mutation rate. But, these new genes will also be easily destroyed via mutation. Genomes which start with very large numbers of genes are disrupted early on in the duplication process by mutation, while those with few genes obtain additional genes through mutation.

To test this explanation, genomes of length  $L_G$  were created completely at random without the use of duplication and divergence. The distribution of these completely randomly generated networks are shown in Figure 5.4(c). As can be seen, this distribution is quite similar to that generated in Figure 5.4(b) lending additional support to the hypothesis that at 5% mutation the network topology becomes effectively randomized.

In the case of no mutations (0% probability of mutation) during the duplication process, we would expect to see a large number of networks which either have zero genes (where there are no 01010101 patterns in the original 32-bit starting string), or have  $2^{\# \text{ of duplications}}$  genes (due to the presence of a 01010101 pattern in the original 32-bit starting string). We wish to obtain a network which shows a topology primarily due to the effects of duplication. Therefore, the distribution of the number of genes in networks generated by duplication and divergence may be used as an estimate of the effect of mutation rate on the network as compared to randomly generated



genomes. Obtaining a network which has a gene distribution approximating a power-law distribution accomplishes this. It is sufficiently randomized so as not to resemble the case of 0% mutation while not being dominated by mutational effects as shown by its lack of similarity to the Gaussian-like distributions shown in Figures 5.4(b) and 5.4(c).

With these considerations in mind, the networks may be examined to determine if their topologies may be considered scale-free and / or small-world.

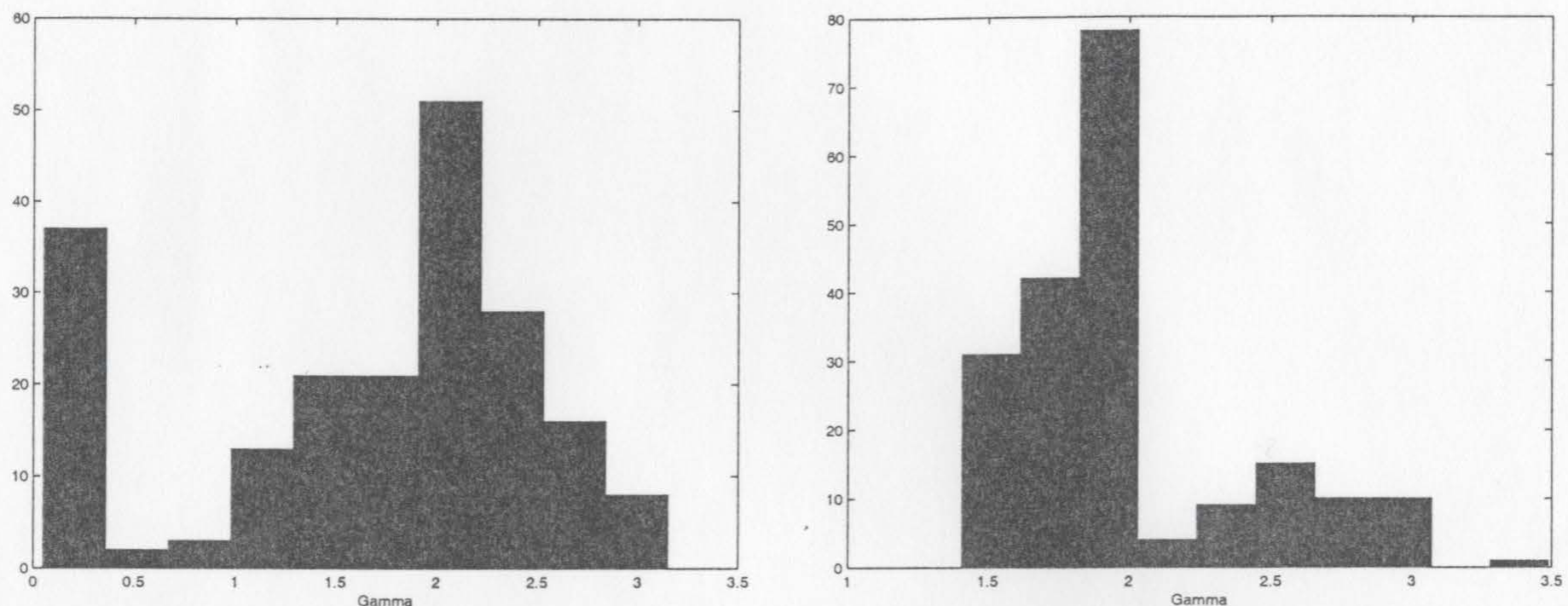
### 5.2.1 Results

The network of gene-protein interactions was parameterized by the threshold value leading to 32 possible networks for each genome (although the case of zero connectivity and full connectivity are neglected). The histograms of the probability of a node being connected with  $k$  components were fitted to the equation  $P(k) = \alpha k^{-\gamma}$  for each threshold value using the sum of least squares method. The threshold value which produced a  $\gamma$  value closest to 2.5 was kept. A large number of networks which have displayed scale-free behaviour exhibit values of  $2 < \gamma \leq 3$  (Goh et al., 2002). Values for the parameter  $\gamma$  characterizing scale-free networks were also calculated for each of the genomes and are shown in Figures 5.5(a) & 5.5(b).

There exist many genomes created by duplication and divergence which may be considered to satisfy the definition of a scale-free network. Figure 5.6 shows an example of one network's connectivity distribution fit to a power-law distribution. The vertex distribution does indeed obey a distribution similar to a power-law (scale-free) distribution.

In Figure 5.5(a), there is a large number of networks whose coefficient  $\gamma$  is close to zero. This would seem to be at odds with the previous statement. However, this





(a) Distribution of values of  $\gamma$  for the best fit of  $P(k) \sim k^{-\gamma}$  with a mutation rate of 1.0%. (b) Distribution of values of  $\gamma$  for the best fit of  $P(k) \sim k^{-\gamma}$  with a mutation rate of 5.0%.

Figure 5.5: Distribution of values of  $\gamma$ .

can be attributed to the fact that since the mutation rate is low, the probability of discovering new promoter patterns through subsequent duplication and divergence steps is not high. Therefore, if there were few promoters in the initial starting string, there will often be few genes in the overall genome. With a small number of genes, the scale-free coefficient  $\gamma$  will often be of small magnitude. In addition, it can be seen from the distribution of  $\gamma$  in Figure 5.5(b) that the majority of the networks created by 5% mutation cannot be classified as scale-free. This again, reinforces the previous finding that a mutation rate of 5% during the duplication and divergence process generates networks that are close to having random connectivity.

To test whether these networks could also be classified as having small-world topology, the clustering coefficient,  $C$ , and the characteristic path length,  $L$ , were calculated and compared to the corresponding metric for a randomly connected network of the same size and vertex degree distribution. The threshold value that produced a network with the smallest absolute difference,  $|L - L_{random}|$ , that also satisfied  $C \gg C_{random}$  were taken to be those most characteristic of the small-world network



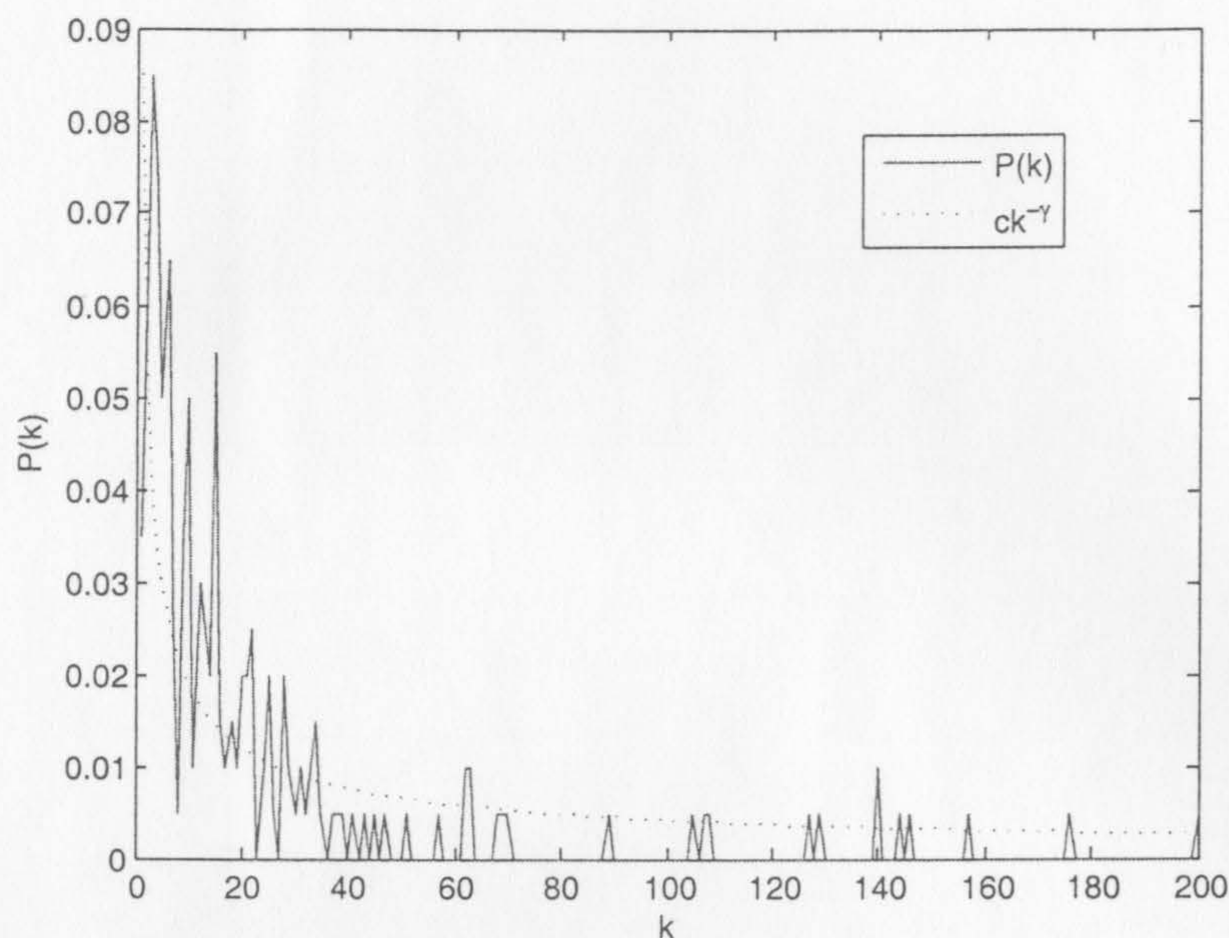


Figure 5.6: Degree distribution of a network generated by duplication and divergence with 1% mutation. The  $P(k)$  axis has been scaled.

topology. The additional constraint that  $L > 1.3$  was also enforced so as to try to exclude graphs that were close to being fully connected.

The distributions for the clustering coefficient and the characteristic path length obtained from the 200 genomes for both rates of mutation are shown in Figures 5.7 & 5.8. From these figures, there exists a threshold at which the interaction network approaches or satisfies the definition of a small-world network topology in the majority of genomes. All graphs which were considered as having scale-free and small-world topology were found in the transition areas of Figure 5.3.

### 5.2.2 Analysis

In light of the results presented in the previous section, an obvious question would be why whole genome duplication creates scale-free and small-world topologies. Firstly, the duplication process, despite being performed directly on the genetic string can be considered to be similar to the mechanism of preferential attachment.

Consider the duplication process on a string which contains multiple genes while



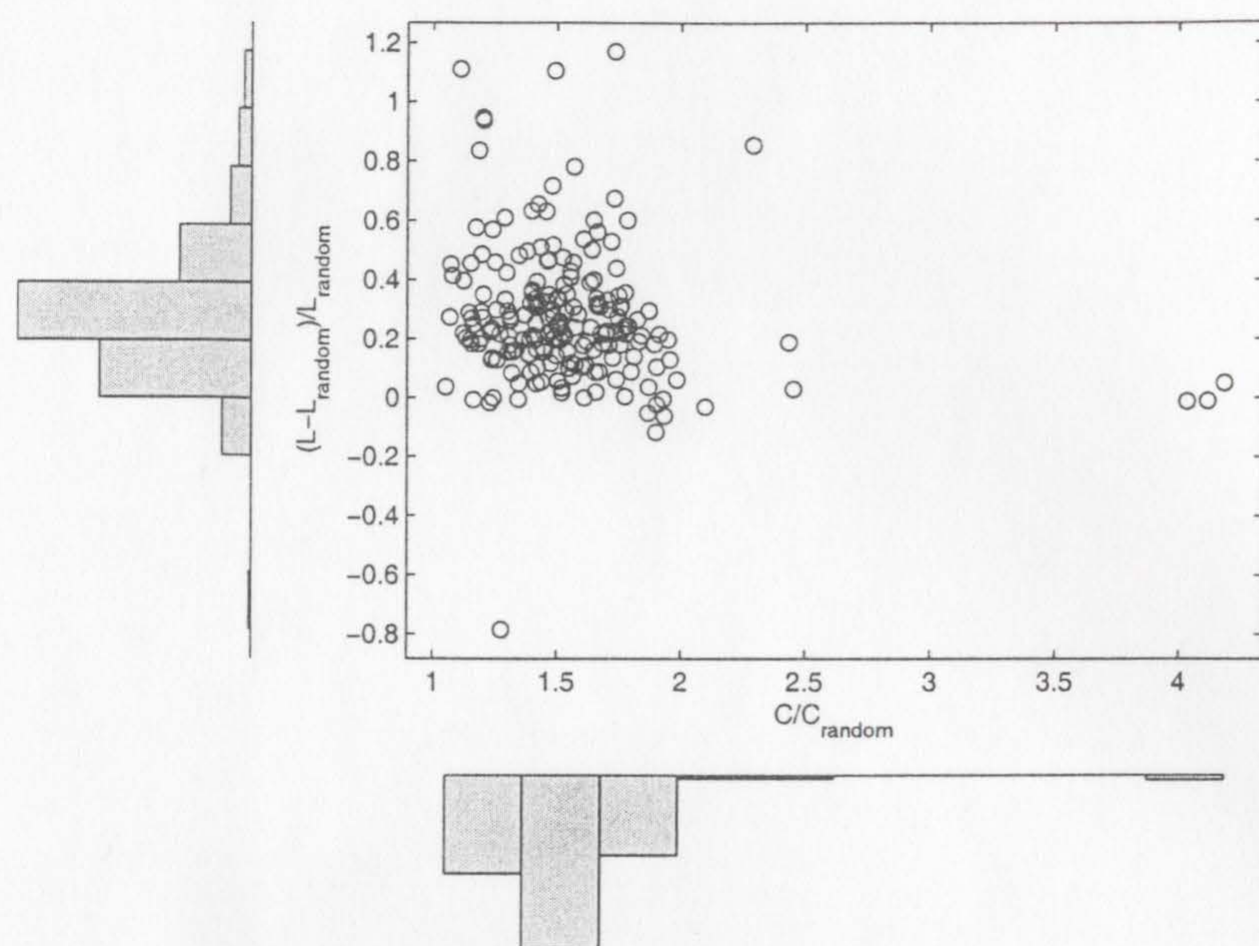


Figure 5.7: Plot of  $\frac{C}{C_{\text{random}}}$  and  $\frac{L_{\text{random}} - L}{L_{\text{random}}}$  for each of the randomly generated genomes (200 genomes) with a mutation rate of 1.0%.

neglecting the effects of mutation. For simplicity, it is assumed that no additional genes are created from a duplication event by joining the end and beginning of one genome string. On the left of Figure 5.9, a network of five gene–protein pairs is shown that proceeds through a single duplication event generating the network shown on the right side.

The more highly connected nodes on the left, nodes 1 and 2 and their copies 6 and 7 (shown in grey) become even more highly connected after a single duplication event. This can again be seen in the third part of the diagram which shows the result of another duplication event. As the number of duplication events increases, the difference in the number of connections between highly connected nodes and less connected nodes increases. This can be thought of as a form of preferential attachment since nodes that are already highly connected will become even more so after subsequent duplication events. Preferential attachment has been shown to be a mechanism which can generate scale-free networks (Barabási and Albert, 1999, Romualdo et al., 2003).

However, this neglects the mechanism of mutation. Mutation may be thought



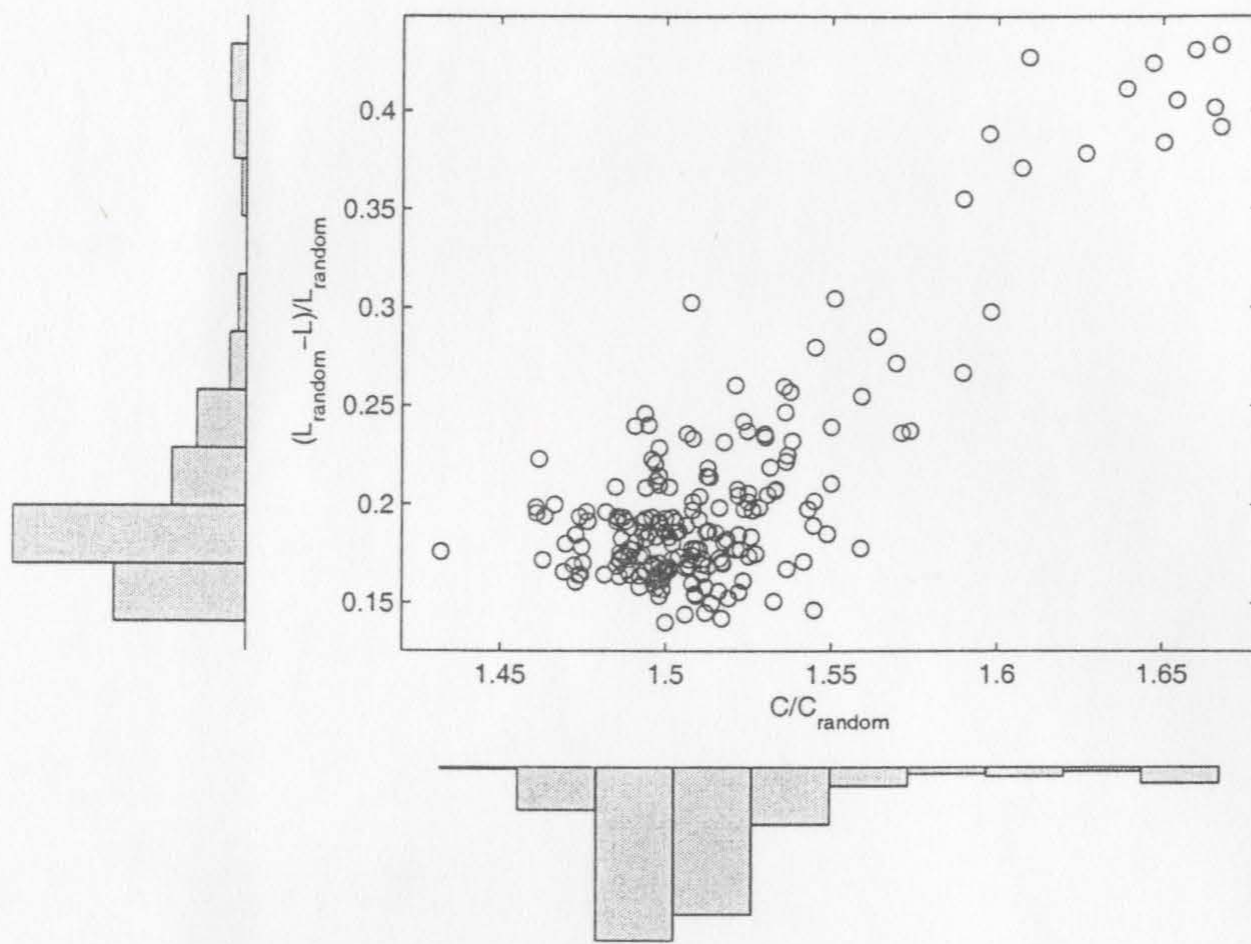


Figure 5.8: Plot of  $\frac{C}{C_{random}}$  and  $\frac{L_{random} - L}{L_{random}}$  for each of the randomly generated genomes (200 genomes) with a mutation rate of 5.0%.

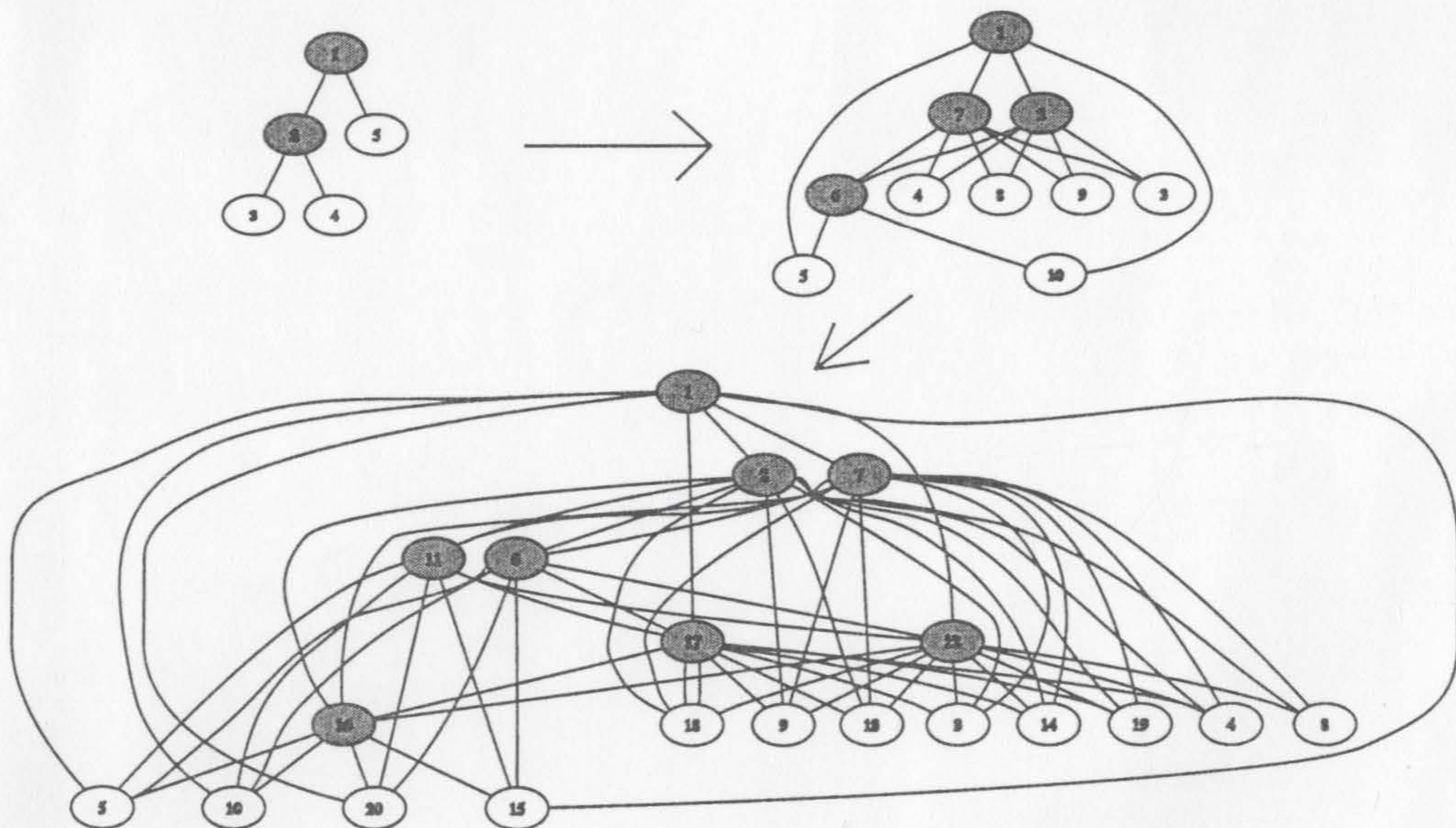


Figure 5.9: An example of the effect of two duplication events. Highly connected (shaded) nodes become even more highly connected (preferential attachment). Each node represents a gene / protein pair; each edge represents an interaction between gene / protein pairs.

of as an operator which reorganizes the network. If mutations should occur on a gene, this may either change the gene-protein pair's binding site, or the generated



protein thus reorganizing a portion of the network. The other possibilities are that mutations may either disrupt the promoter pattern in effect deleting a gene–protein pair from the network, create a new gene–protein pair by creating a new promoter site, or are neutral. Regardless, it was shown in the previous section that the topology of the network as measured by the number of genes in the system is dominated by the effects of duplication, not divergence. Thus, we can be confident that the scale-free distribution observed is due to the duplication mechanism, acting similarly to preferential attachment.

How can the small-world topologies found in the ARN model be explained? If we examine the definition of a small-world network more closely, it colloquially states that a network is highly clustered but that there are many links between these clusters which effectively reduce the overall diameter of the network. Frequently, hubs also appear in small-world networks (Watts, 2003). It is clear that hubs can appear in the ARN model through the duplication process (analogous to preferential attachment to more highly connected nodes). However, because of the way the duplication process functions (assuming no mutation), the maximum distance<sup>1</sup> between any two nodes before and after a duplication remains constant. This occurs because the duplication step effectively makes a copy of all nodes and all edges simultaneously. It is self-evident that the maximum distance between any two nodes in only the original graph and the copied portion of the network are the same (if we discount the edges which connect the original nodes with the copied nodes). This shows that the path length between any two nodes in the original graph is the same as in the copy.

If we replace any node in the original graph (nodes 1, 2, and 3) with its copy (nodes 1', 2', and 3') and its associated edges to the original graph, the overall topology remains identical. This shows that the path length between any two nodes in the

---

<sup>1</sup>The number of edges traversed to get from node “a” to “b”



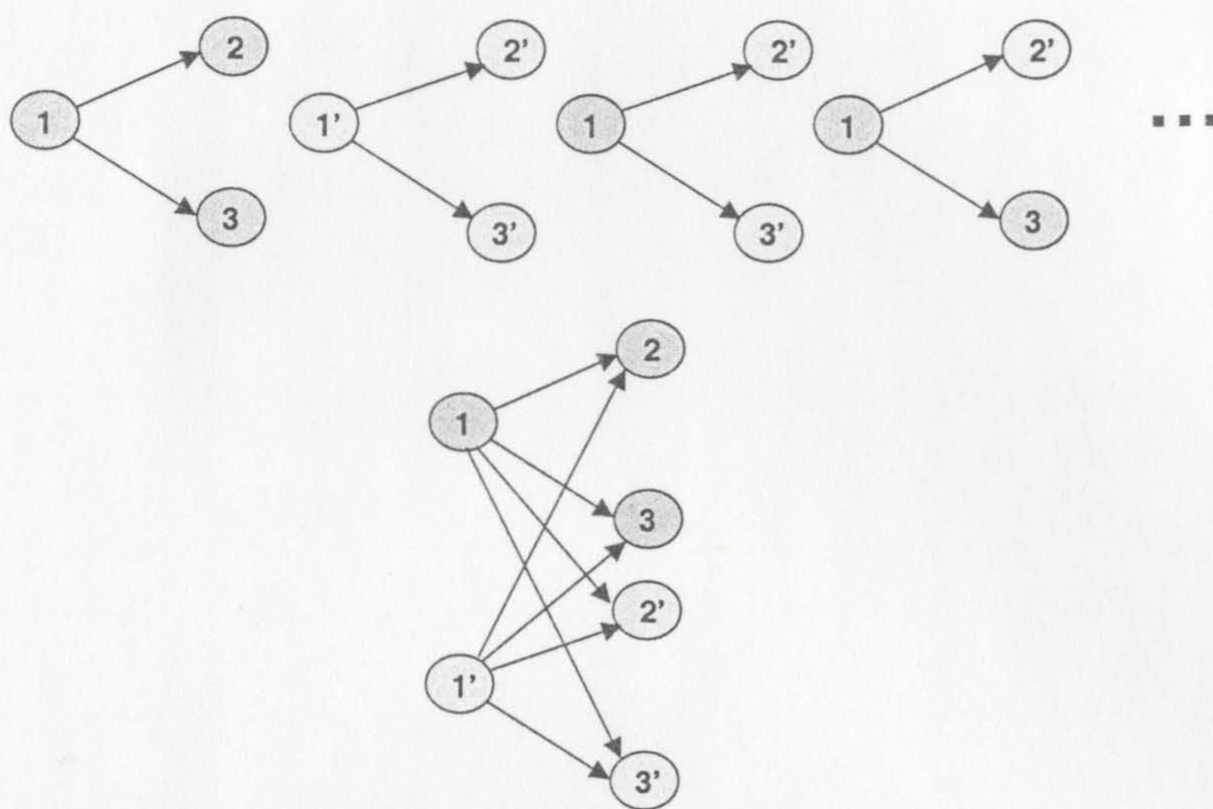


Figure 5.10:

Demonstrates that any of the nodes in the original topology can be replaced with its copy without changing the topology and vice versa. Therefore, the maximum path length remains constant.

original graph is the same as the path length between either of the nodes in the original graph and a copy of the other node in the copied graph. This shows that the maximum path length is invariant to duplication (and thus generally remains small). This is shown in Figure 5.10. Therefore, the average path length will always be bounded by the maximum path length which we know will never increase. As the network grows via the duplication process, its characteristic path length will grow much more slowly if at all due to mutations.

It is also evident that the clustering coefficient of the network is also quite high as a result of the duplication process. Because of the regularity of the connection patterns, nodes in the network remain highly connected and in fact increase in connectivity with each duplication event. Mutation only serves to perturb the topology effectively partially randomizing some of the edge connections in the graph. Thus, the formation of small-world type topologies can be consistent with the network creation method of whole genome duplication and divergence.



## 5.3 Network Motifs in the ARN Model

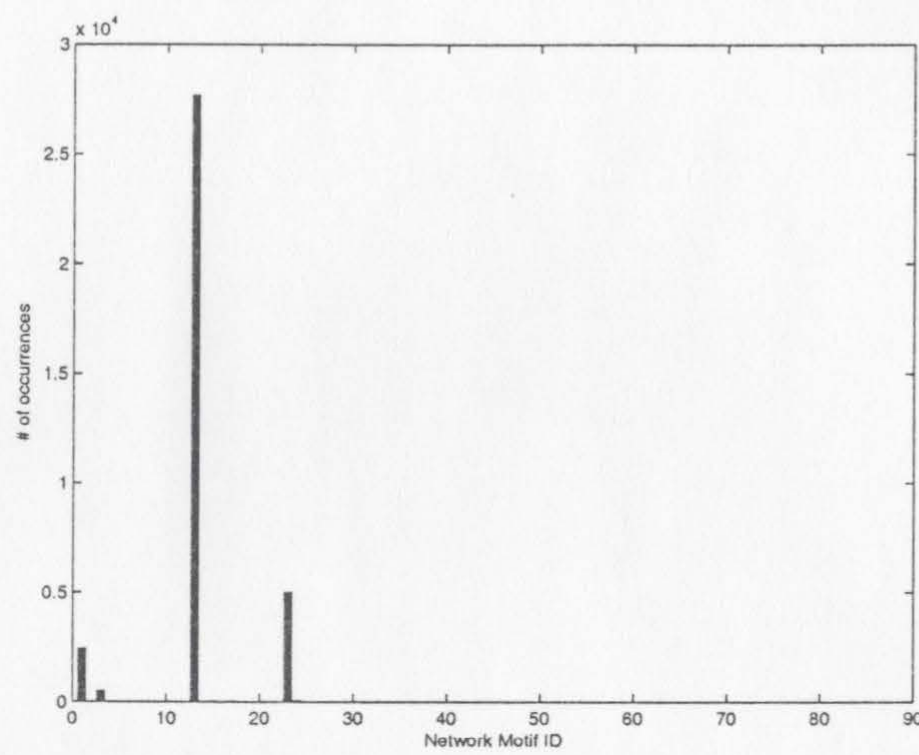
Analysis of network motifs has become one method with which to study the topology of transcriptional regulatory networks. Methods are mainly based on searching for connection patterns among small numbers of nodes. Here, a network motif finding algorithm is applied to the artificial regulatory network model previously introduced and compared to results obtained on the natural regulatory networks of *Escherichia coli* and *Saccharomyces cerevisiae* in addition to being compared to randomly generated control networks. The high frequency of certain network motifs detected in natural systems can be found in artificial systems as well, provided they are generated by a gene duplication and divergence process. This leads us to suggest that the actual frequency distribution of motifs (“motif fingerprint”) in natural regulatory networks could be at least partially a consequence of the process of network generation rather than of subsequent evolutionary selection. A discussion of the network motif algorithm implementation can be found in Appendix C.2.

### 5.3.1 Results

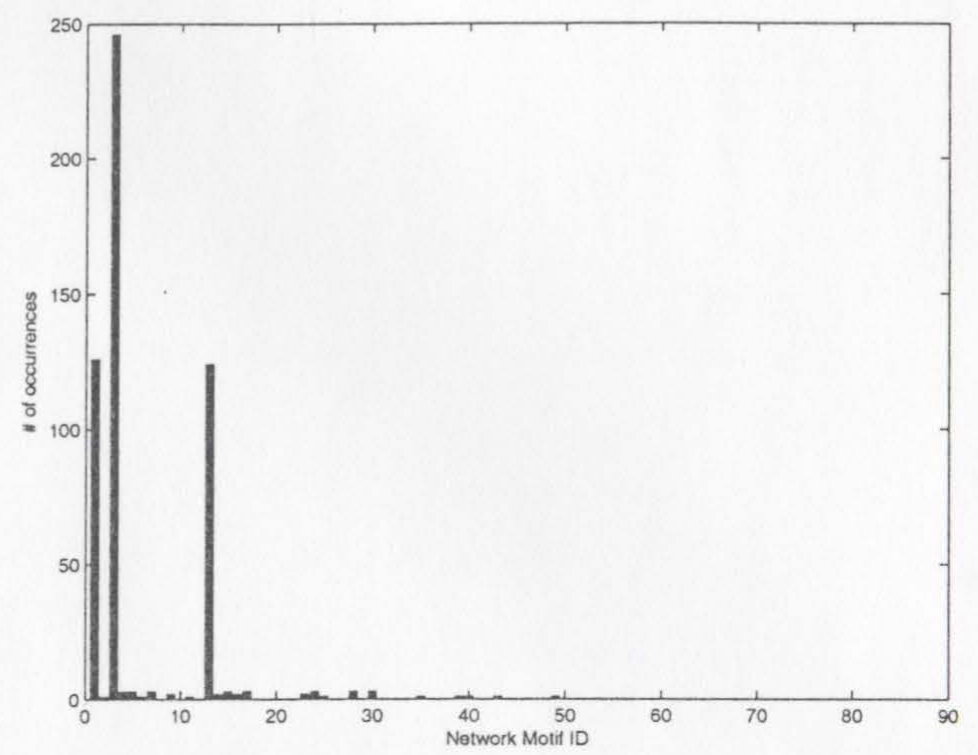
The subgraph finding algorithm was applied to 800 instances of the artificial regulatory model generated by the duplication and divergence process. As a control, it was additionally applied to 800 networks whose genomes were generated randomly (by choosing the full number of bits at random). Results of applying the subgraph counting algorithm to the two cases are shown in Figures 5.11(a) and 5.11(b). For both methods of network generation, the genome length was set at 131072 (12 duplication events in the case of duplication and divergence). For networks generated by duplication and divergence, the mutation rate was set at 1% using the same justification presented earlier in this chapter.



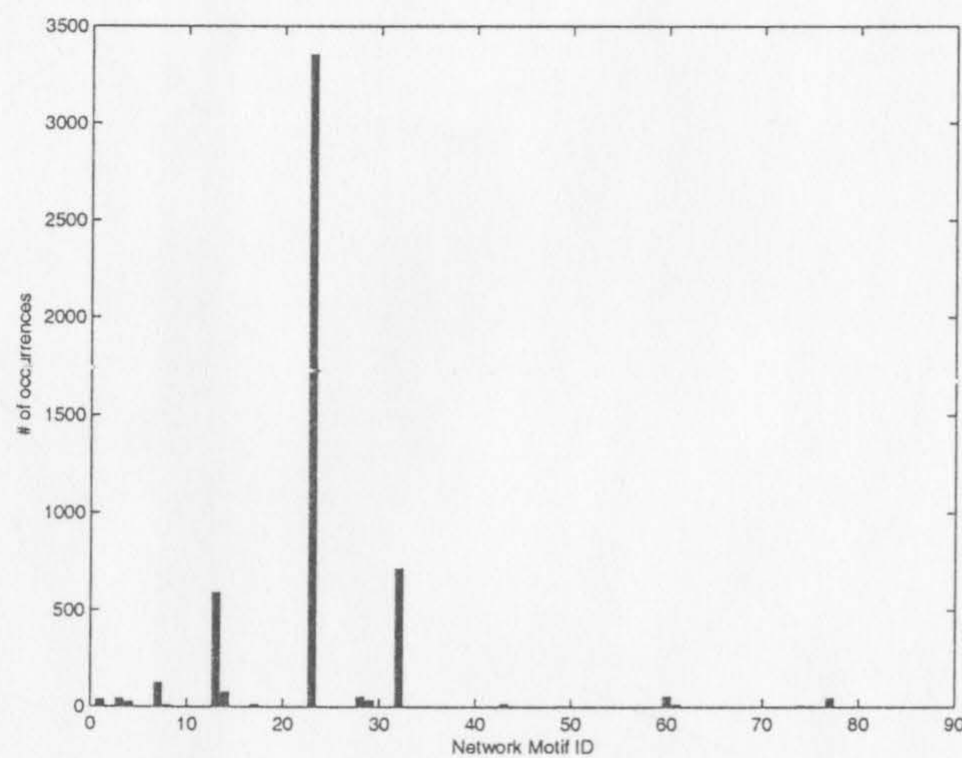
In both cases, the threshold had to be determined. The ratio of the number of edges to the number of vertices for the two natural regulatory networks was approximately 2 to 1. Therefore, in our artificial regulatory network framework, the threshold was chosen by iteratively raising the threshold until the network generated had a ratio that was equal to or less than 2 to 1.



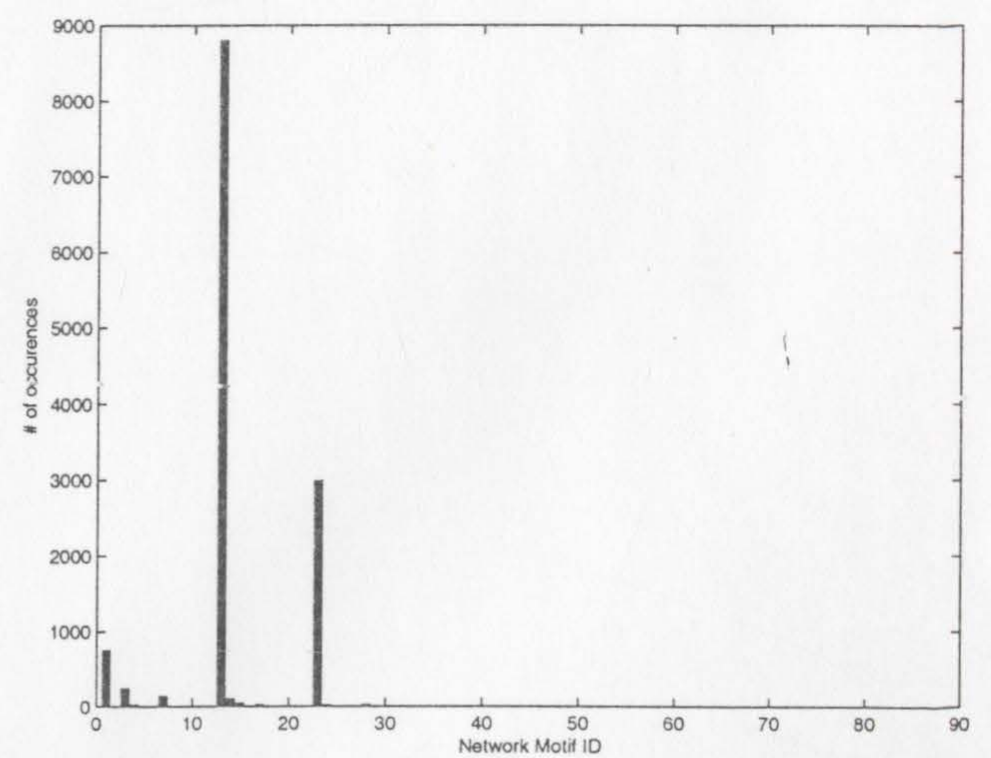
(a) Average frequency of occurrence for subgraphs of size three in 800 instances of the artificial regulatory network model generated by a duplication and divergence procedure.



(b) Average frequency of occurrence for subgraphs of size three in 800 randomly generated instances of the artificial regulatory network model.



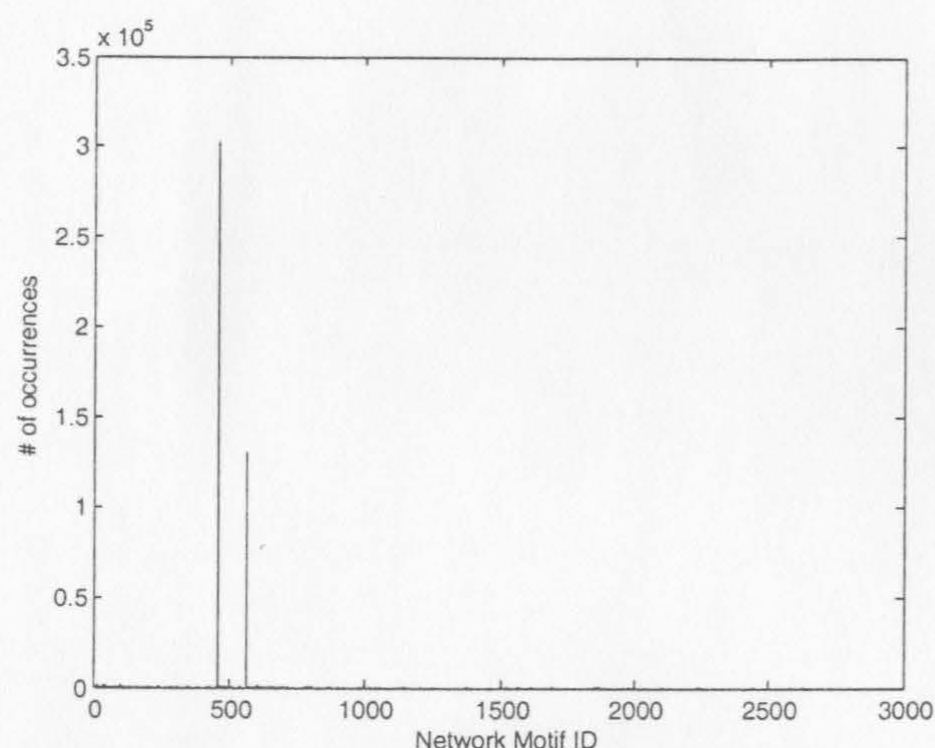
(c) Frequency of occurrence for subgraphs of size three in the transcriptional network of *Escherichia coli*.



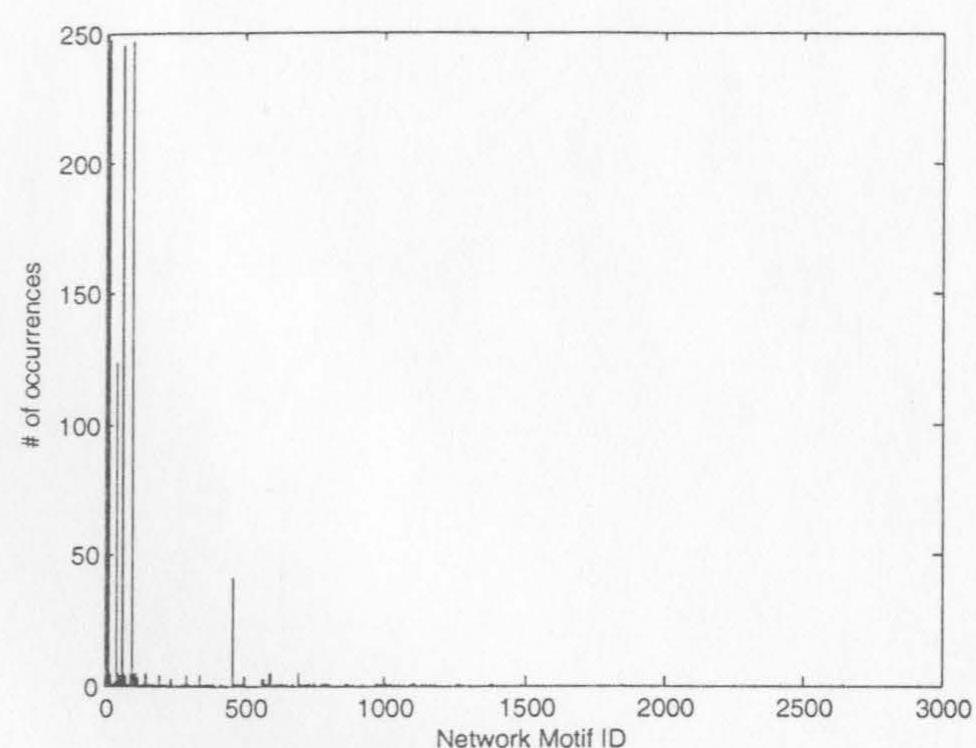
(d) Frequency of occurrence for subgraphs of size three in the transcriptional network of *Saccharomyces cerevisiae*.

Figure 5.11: Frequency of occurrence for subgraphs of size three.

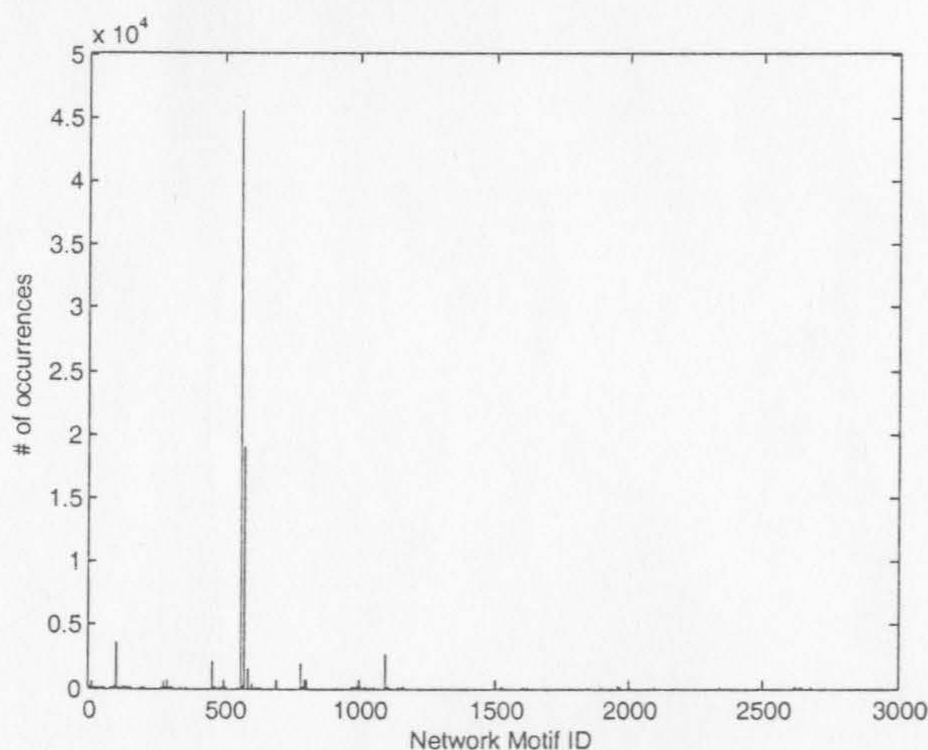




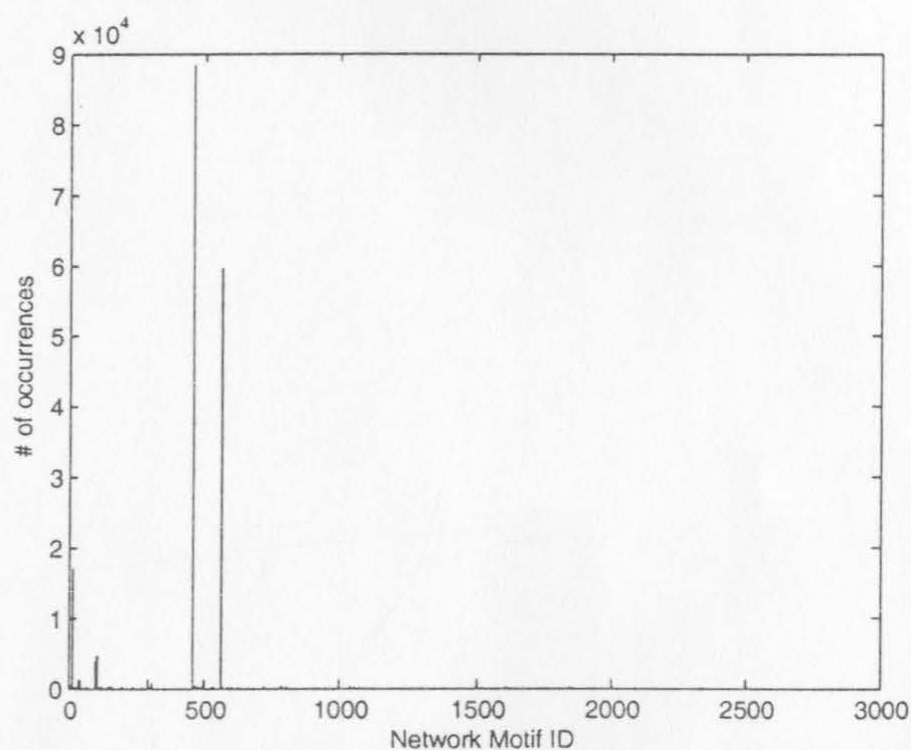
(a) Average frequency of occurrence for subgraphs of size four in 200 instances of the artificial regulatory network model generated by a duplication and divergence procedure.



(b) Average frequency of occurrence for subgraphs of size four in 200 randomly generated instances of the artificial regulatory network model.



(c) Frequency of occurrence for subgraphs of size four in the transcriptional network of *Escherichia coli*.



(d) Frequency of occurrence for subgraphs of size four in the transcriptional network of *Saccharomyces cerevisiae*.

Figure 5.12: Frequency of occurrence for subgraphs of size four.

This was then compared to the results of applying the algorithm to two natural transcriptional networks<sup>2</sup>, *Escherichia coli* (Thieffry et al., 1998, Shen-Or et al., 2002) and *Saccharomyces cerevisiae* (Costanzo et al., 2001). The results of application of

<sup>2</sup>The network topologies for these transcriptional regulatory networks was obtained from Uri Alon at <http://www.weizmann.ac.il/mcb/UriAlon/>.



the network motif algorithm to these two networks can be seen in Figures 5.11(c) and 5.11(d). It can be seen in Figures 5.11(a) – 5.11(d) that the most frequent natural subgraphs (ID 22 and ID 12) are both well represented in duplication and divergence type artificial networks whereas only one of them can be detected in fully random networks.

The subgraphs counts for subgraphs of size three and four for all the types of regulatory networks investigated in this thesis are presented in Appendices F and H. For the artificial networks, average numbers of counts are shown, whereas for the natural regulatory systems only one network each is investigated.

### 5.3.2 Analysis

Using the sum of square error (SSE) criterion, the similarity between the distributions of subgraphs between the four types of networks was calculated. The similarity is shown for both three and four node subgraphs in Tables 5.1 and 5.2.

	D/D	Rand	EColi	Yeast
D/D	0	1.5348	1.0844	0.0072
Rand	1.5348	0	2.2392	1.4886
EColi	1.0844	2.2392	0	1.1693
Yeast	0.0072	1.4886	1.1693	0

Table 5.1: Sum of square error (SSE) between the distributions of subgraph counts (for subgraph size three) for the four types of networks examined. Each distribution has been normalized such that the maximum count of any individual subgraph is 1.0.

	D/D	Rand	EColi	Yeast
D/D	0	5.3093	1.4227	0.0984
Rand	5.3093	0	5.6148	5.1497
EColi	1.4227	5.6148	0	1.2356
Yeast	0.0984	5.1497	1.2356	0

Table 5.2: Sum of square error (SSE) between the distributions of subgraph counts (for subgraph size four) for the four types of networks examined. Each distribution has been normalized such that the maximum count of any individual subgraph is 1.0.

The network distributions obtained from duplication and divergence are quite similar to that of *Saccharomyces cerevisiae* for subgraph sizes of both three and four



according to the SSE criterion. In contrast, it can be seen in the tables that the distributions of the randomly generated networks were not similar to any of the three other networks investigated. In fact, it can be seen from the tables that networks created by duplication and divergence and the regulatory networks of *Escherichia coli* and *Saccharomyces cerevisiae* are all more similar to each other than to the randomly generated networks.

As gene duplication is considered a more important mechanism of evolution in eukaryotes than in prokaryotes, it is interesting that the duplication and divergence networks are more similar to the eukaryotic *Saccharomyces cerevisiae* rather than the prokaryotic *Escherichia coli*. This might suggest that the topology has been shaped by duplication events in *Saccharomyces cerevisiae*'s evolutionary history. It is in fact been suggested by Teichmann and Babu (2004) that over 90% of eukaryotic genes are created by gene duplication. Regardless, it is striking how similar the distributions of subgraphs are for these three networks as compared to the randomly created topologies.

In addition, we can investigate the individual subgraphs which are well represented in these networks. It can be seen from Figures 5.11(a), 5.11(c) and 5.11(d) that IDs 22 and 12 are present in quite high numbers. These motifs correspond to the so-called single input module (Milo et al., 2002). This is also the case when examining subgraphs of size four in Figures 5.12(a), 5.12(c) and 5.12(d) where network motif IDs 459 and 563 are well represented. In counts of both three and four node subgraphs, the single input modules were not well represented in randomly created graphs.

We have observed that the single-input module is present in the natural networks and the network created by duplication and divergence. A valid question would be how the single-input module might be created by duplication and divergence? We can examine the effect of duplication on the simplest of gene interactions, where one



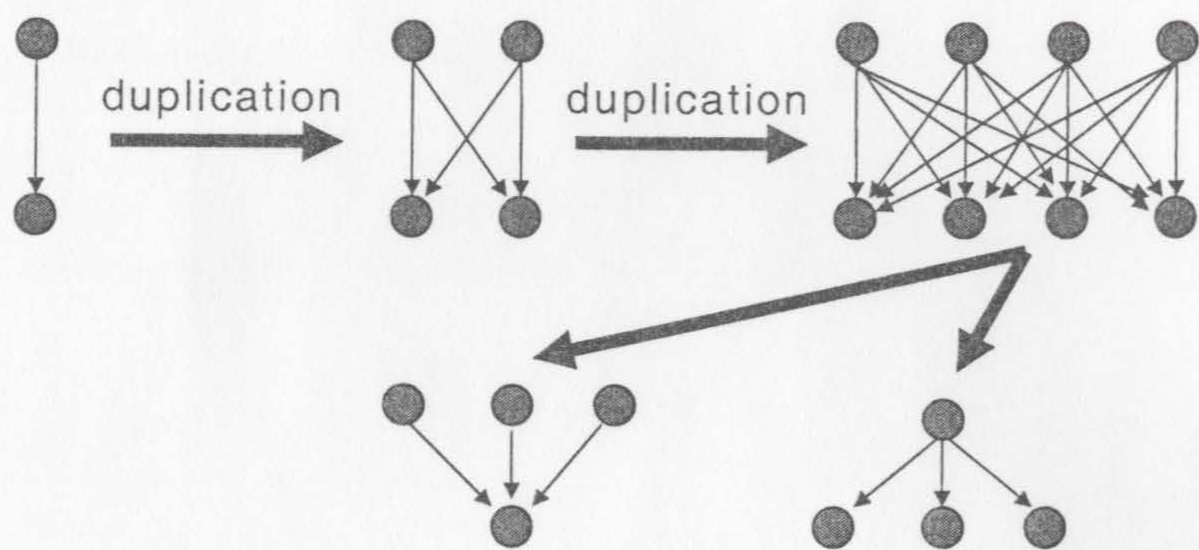


Figure 5.13: The effect of whole genome duplication on the simplest possible interaction between two genes.

gene has a regulatory influence on another. If these genes and their connections are duplicated we can obtain the so-called single input module network motif.

Figure 5.13 shows the effects of two duplications on the simplest of regulatory influences. This should create two types of subgraphs with equal probability, the single-input module and the so-called single-output module as shown in the figure. However, this is not the case as can be seen when examining the motif counts for both the natural and artificial networks. This is a natural consequence of the duplication and divergence process and has been studied by Leier et al. (2005, in preparation).

## 5.4 Conclusion

Investigations on the topological properties of an artificial regulatory network model have been presented. The construction of such a network using a simple whole genome duplication process directly on a genetic-string representation of the genome produces a network construction scheme similar to preferential attachment. The addition of a mutation operator introduces a kind of rewiring of the network topology by changing activation / inhibition sites, creating / destroying gene-protein pairs and changing the configuration of proteins. Examining networks generated in this way by varying the threshold at which genes and proteins may interact shows that many of these regulatory networks display the characteristics of small-world and scale-free network



topologies with some regularity.

This assumes that duplication proceeds by duplicating the whole genome – an event which occurs relatively rarely in nature (Nadeau and Sankoff, 1997, Wolfe and Shields, 1997). That the networks generated are scale-free is in agreement with Chung et al. (2003) and van Noort et al. (2004). It was also found that the *Saccharomyces cerevisiae* co-expression network displays small-world topology (van Noort et al., 2004).

The data presented regarding the distribution of network motifs among the artificial and natural networks might convince us that there might be a relationship between them. No evolutionary selection pressure has been applied to the artificial systems. Thus, it can be stated that the distribution outcome is a reflection of the mechanism of its generation rather than a result of evolutionary pressures as are the case in natural networks. Perhaps it may be the case that the motif distributions in these natural networks are in part the result of other organizing forces such as duplication and divergence (although evolutionary pressures are certainly responsible for fine-tuning of distributions).

As was previously noted, there is conflicting evidence in the literature regarding the conservation of network motifs over evolutionary time (Wuchty et al., 2003, Babu et al., 2004). In addition, Babu et al. (2004) and Teichmann and Babu (2004) suggest that network motifs are not created by duplication events but are built by incremental evolution of gene interactions. In support of this conjecture, it was suggested by Conant and Wagner (2003) that network motifs are found through convergent evolution – not through any duplication processes.



## Chapter 6

# Evolving Dynamics in the ARN Model

In the previous chapter, the topology of the ARN model was investigated. However, topology is only one of the aspects of a genetic regulatory network. It could be argued that topology is what gives rise to dynamics, but regardless, it is the actual dynamics of the network that give rise to the myriad of functions observed in natural systems. This chapter examines the dynamics of the ARN model from the standpoint of attempting to evolve simple time series.

In addition to the interest from biology, the artificial life community has also been studying genetic regulatory networks (Reil, 1999, Bongard, 2002, Banzhaf, 2003b, Bongard and Pfeifer, 2003, Hotz, 2003, Watson et al., 2003, Willadsen and Wiles, 2003, Hallinan and Wiles, 2004). As such, features of regulatory networks have also been used in the context of function optimization by Bongard (2002), Bongard and Pfeifer (2003) and Watson et al. (2003). Obtaining arbitrary functions through evolutionary means for the purpose of model optimization has been previously performed for flying (Augustsson et al., 2002), locomotion (Dittrich et al., 1998) and the inference of



differential equations (Cao et al., 2000).

However, previous models of regulatory networks primarily use Boolean representations of network dynamics (Reil, 1999, Watson et al., 2003, Willadsen and Wiles, 2003, Hallinan and Wiles, 2004). Here we show that an ARN model using differential equations can also display simple dynamic behaviours which may be selected by evolution. Other ideas relating to genetic transcription have also previously been used in function optimization such as genetic-code transformations (Kargupta and Ghosh, 2002), gene expression (Kargupta, 1996, Eggenberger, 1997), gene signalling (Goldberg et al., 1989) and diploidity (Yoshida and Adachi, 1994).

Thus, by attempting to evolve an arbitrary time series in the ARN model, some enquiries on the evolvability of the ARN model can be performed with some possible relevance to the evolvability of natural systems. The types of analysis and search mechanisms relevant to such processes could also be important to the field of synthetic biology where synthetic genetic regulatory networks have been evolved *in vivo* toward dynamics such as oscillations (Hasty, 2002, Yokobayashi et al., 2002), *in numero* (François and Hakim, 2004) and *in silico* (Mason et al., 2004). Such an investigation also provides a framework where the interplay between network dynamics, evolution and topology can begin to be investigated. Portions of the work presented in this chapter have been previously published in Kuo et al. (2004).

## 6.1 Extracting a Signal from the ARN Model

Simulation of the ARN model presented in Section 3.4 gives the dynamics of the protein concentrations in the system. However, the system has no assigned semantics – the protein concentrations have no meaning outside the system (they perform no internal or external cellular function other than regulation). Additionally, since the



protein concentrations are limited to sum to 1 (i.e.  $\sum c_i = 1$ ), generation of functions in which the protein concentrations do not sum to 1 are excluded.

Therefore, in order to use the ARN framework to obtain more arbitrary dynamics, a mapping is required. To this end, an additional 64-bit sequence is randomly selected along the genome as a binding site for the desired output function. The first 32-bits specify a transcription factor binding site representing an inhibition site while the second 32-bits specify a transcription factor binding site for activation. Remember, that proteins acting as transcription factors can bind to transcription factor binding sites in order to influence the creation of a protein from an adjacent gene. The proteins generated by the ARN are free to bind to these two additional regulatory sites. The levels of activation and inhibition are calculated in the same way as in Equation 3.10.

However, instead of calculating a “concentration” of this site (which generates no protein of its own), the activity at this site is simply summed and used directly as an output function,  $s(t) = \sum_i (e_i - h_i)$ .

Subsequent normalization of  $s(t)$  to between  $-1$  and  $1$  generates the dynamics of the specific genome. Without this normalization step, it is difficult to match the scaling of any desired dynamics. However, since this scaling is effectively arbitrary depending only on the specific desired dynamics, this is not a problem.

Thus, the additional binding sites added to the genome may be thought of as a method with which to extract dynamics from the changes in concentrations of the proteins in the ARN model. This can be visualized as being a network like the ones presented in Figures 5.2(a) and 5.2(b) except where each protein is linked to an additional node representing the new inhibition / activation site (but does not generate a protein of its own). Additional inhibition / activation sites may also be added to the genome for the extraction of additional signals.



## 6.2 Evolutionary Strategies

The evolutionary strategies approach to artificial evolution was selected in order to evolve the simple dynamics in the ARN model. Such an approach is easily implemented and has been used previously for evolving genetic circuits in *in numero* and *in silico* synthetic biology (François and Hakim, 2004, Mason et al., 2004).

Specifically, a  $(\mu + \lambda)$ -Evolutionary Strategy (ES) was used where  $\mu$  and  $\lambda$  represent the number of parents and offspring respectively. The “+” indicates that selection occurs over both the parents and their offspring. Therefore, in one generation (or step) of the algorithm,  $\lambda$  offspring are generated from the  $\mu$  parents. Then, the best  $\mu$  individuals of the group of this generation’s parents,  $\mu$ , and offspring,  $\lambda$ , are taken to form the parents of the next generation. An alternative to the “+” selection strategy is the use of the “,” strategy. In this selection strategy, only the best  $\mu$  of the  $\lambda$  offspring are selected to survive to become parents in the next generation of the algorithm. Regardless of the selection strategy, offspring are created by taking each of the parents and applying a mutation operator on them which is problem dependent. In general, a recombination operator analogous to crossover in eukaryotic organisms may also be used but is omitted here. A good introduction and review of work in the field of evolutionary strategies can be found in Beyer and Schwefel (2002).

## 6.3 Optimization and Simulation Details

In order to evolve solutions,  $s(t)$ , a simple  $(50 + 100)$ -Evolutionary Strategy (ES) is used (Beyer and Schwefel, 2002). Pseudocode for this process is presented as follows:



---

**Algorithm 1:** Pseudocode for the evolutionary strategies algorithm.

---

```
initialize 50 random individuals;
evaluate the population's fitness;
while population has not reached convergence do
  for each member of the population do
    for perform this twice for each population member do
      create a copy of the population member and mutate it;
      evaluate this new mutated member of the population;
    end
  end
  select the best 50 individuals from the group of parents and offspring;
end
```

---

Genomes were generated by 10 duplication events per genome subject to 1% mutation leading to individual genomes of length  $L_G = 32768$ . It was previously shown in Section 5.2 that a mutation rate of 1% during the duplication and divergence process is sufficient to “rewire” parts of the topology of the network without making it completely random (Kuo and Banzhaf, 2004).

The number of genes in each genome is given by the number of promoter patterns present as was previously defined in Section 3.4. Each generation, 100 new individuals are created from the current population using a 1% single-point (bit-flip) mutation (i.e. on average, 328 mutations per genome). The fitness of these solutions was calculated and the best 50 of 150 (parents + children) proceed to the next generation. ES was stopped when the best solution found was not improved upon for 250 generations.

The objective is to minimize the fitness function calculated as the mean square error (MSE) between the desired function and the evolved function. The following cases were examined and are shown in Figure 6.1:  $f(t) = \sin(t)$  (Case #1),  $f(t) =$



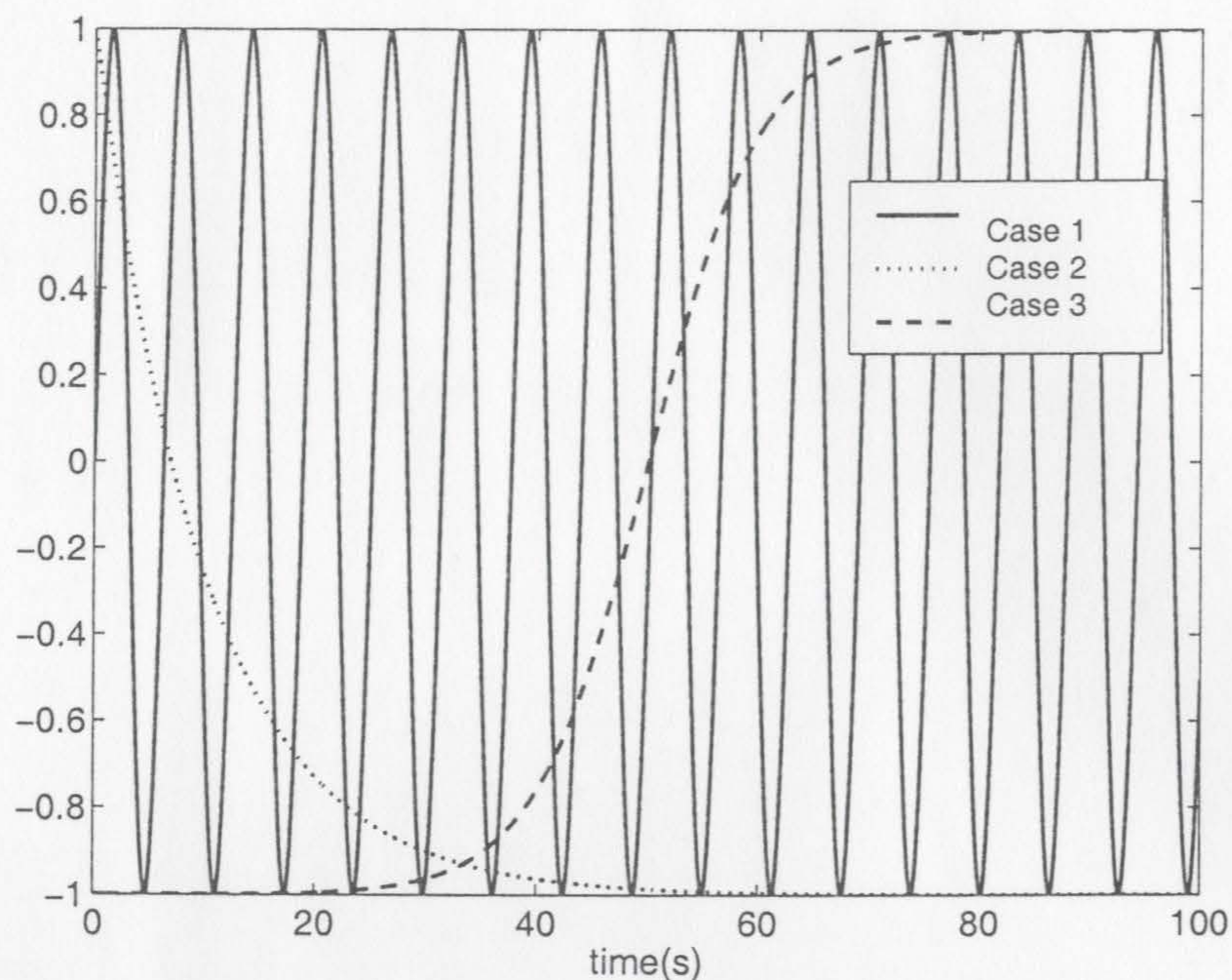


Figure 6.1: Plot of the three fitness cases. Each run aims to match the dynamics of one of these three cases (ten runs were performed for each case).

$2 \exp(-0.1t) - 1$  (Case #2) and  $f(t) = \frac{2}{1 + \exp(-0.2t + 10)} - 1$  (Case #3).

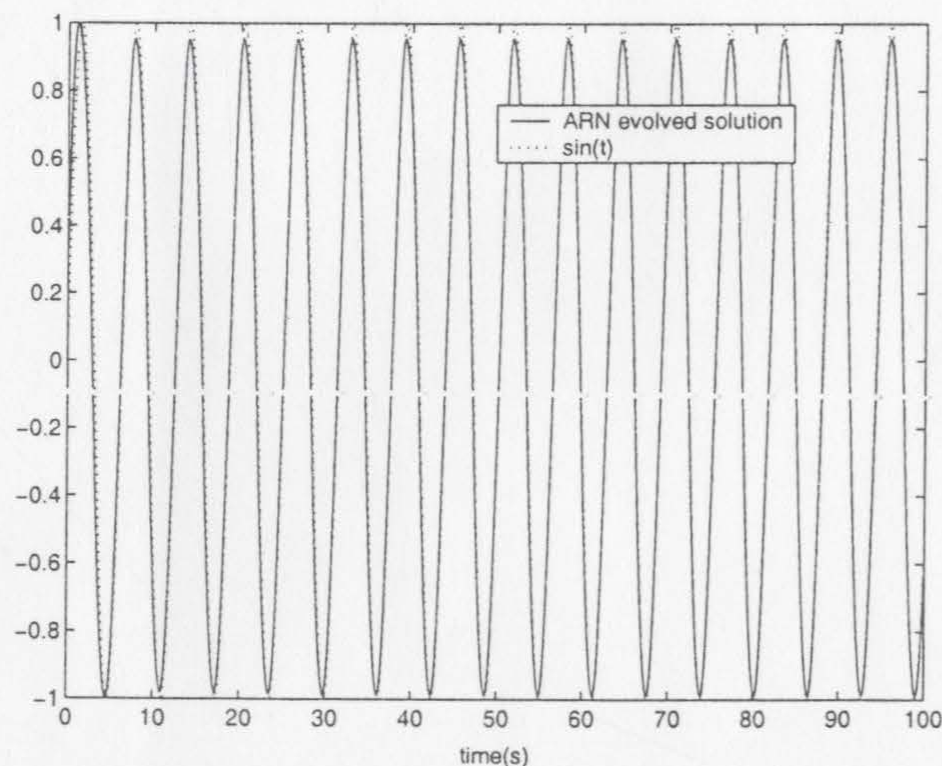
All solutions were generated with a time step,  $dt = 0.1s$ . The initial protein concentrations (the initial conditions for the differential equation) are set to be  $\frac{1}{\#of genes}$ . In addition, the first 100 time steps (10s) are ignored. This is done in order to exclude the startup dynamics of the model. Thus, for calculation of the fitness function, the normalized output generated by the ARN model from time  $t = 10 \dots 110s$  is compared with the fitness case  $f(t)$  from time  $t = 0 \dots 100s$ . The differential equation model is solved using a simple integrator, in this case Euler's algorithm. Normally, the use of such a naïve integrator can cause significant numerical error. However, due to the simplicity of the differential equations which are simple linear functions and the small time step of  $dt = 0.1s$ , there are no problems with either numerical stability of the algorithm or problems with singularities or nonlinear behaviours.



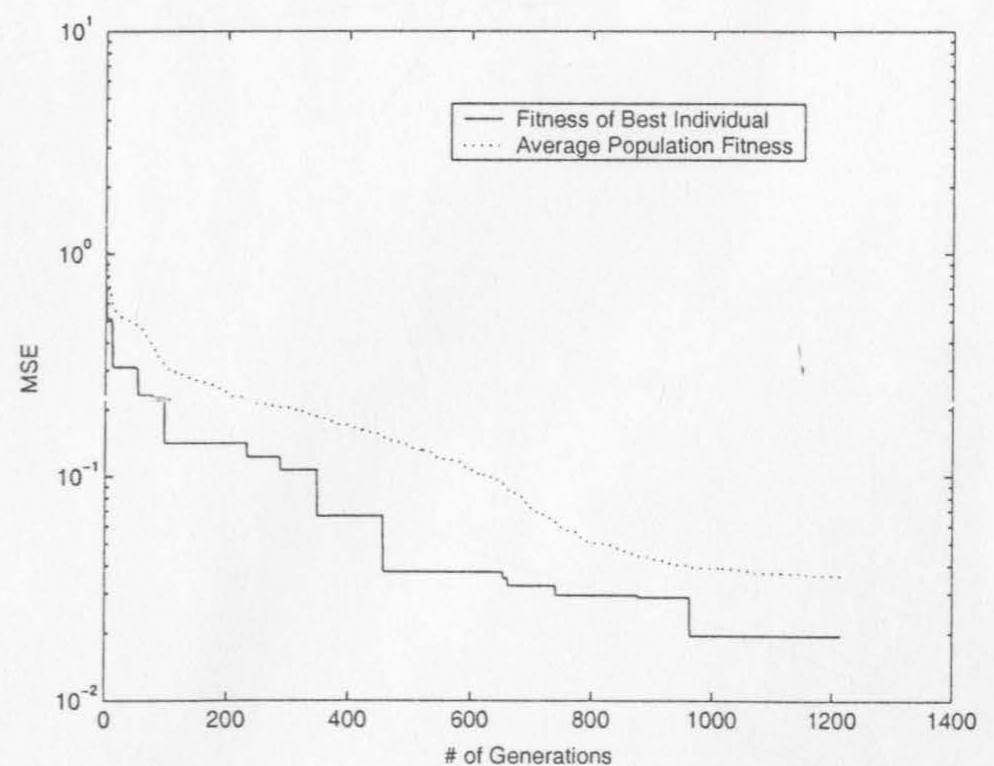
## 6.4 Results

Tables 6.1, 6.2 and 6.3 summarize the results of 10 evolutionary runs each for the three fitness cases. Figures 6.2(a), 6.3(a) and 6.4(a) show the actual function generated by the best individual of each run for the three fitness cases. Figures 6.2(b), 6.3(b) and 6.4(b) show the progress of the best evolutionary run for each fitness case.

It is clearly shown that the ARN model accurately generates dynamics approximating the sinusoid (Figure 6.2(a)), the exponential (Figure 6.3(a)) and the sigmoid (Figure 6.4(a)) functions with good accuracy for all runs. In all fitness cases and evolutionary runs, the MSE calculated was less than 0.00588654. Additional support for the success of these simulations can be seen in the final population fitness averages shown in Tables 6.1, 6.2 and 6.3. The average population fitness values (MSE) are relatively small with low standard deviation. This indicates that the population is such that all or virtually all individuals when simulated generate functions that closely approximate the respective objective functions.



(a) Plot of the best solution (run #8) compared to the ideal solution for Case #1 (sinusoid). The MSE is 0.000151746.



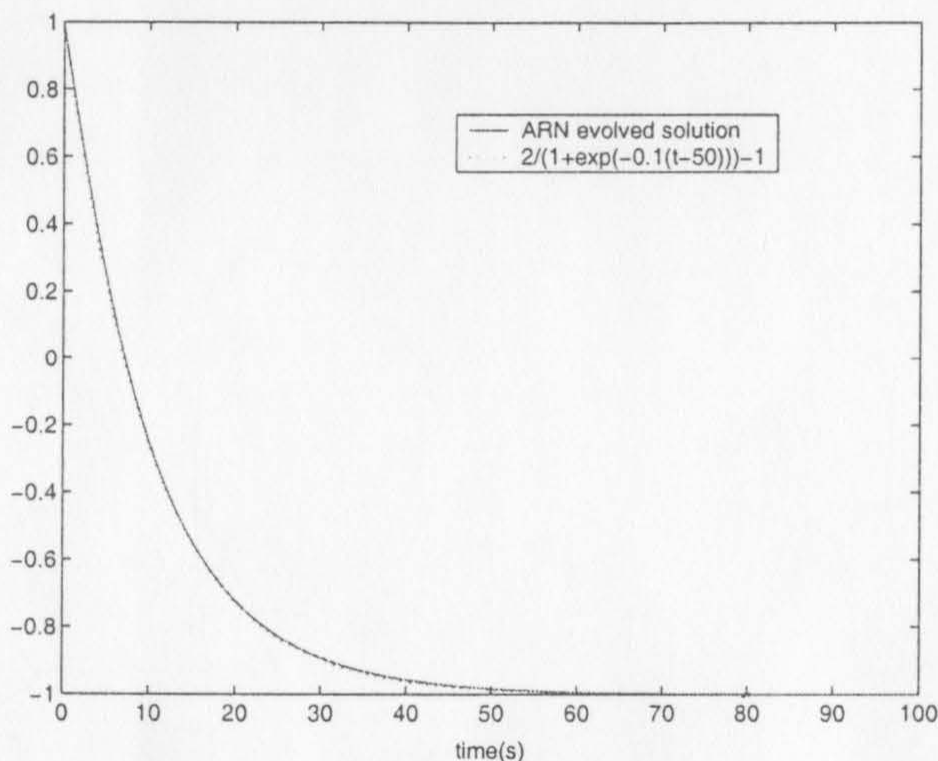
(b) Plot of the fitness of the best solution (run #8) and the average fitness using (50 + 100)-ES for Case #1 (sinusoid).

Figure 6.2: The best solution of 10 runs on Case #1.

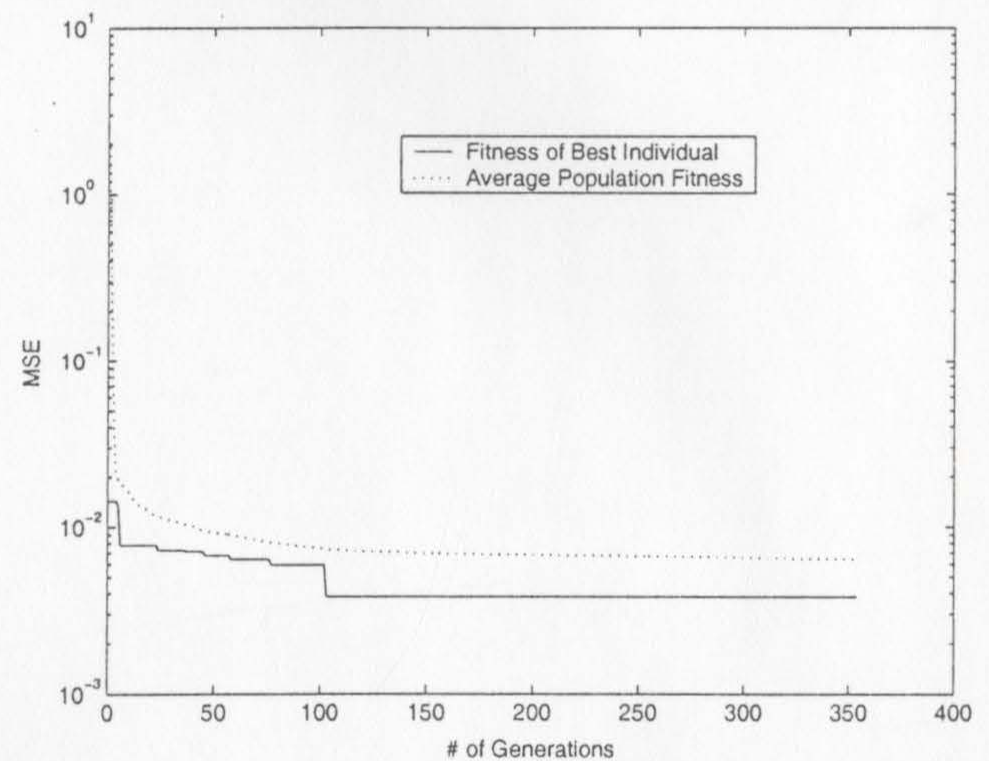


Run #	Best MSE	#Gens.	#Genes	Avg. MSE(Pop.)	Avg. #Genes (Pop.)
1	0.001445217	731	47	0.00287(7.7e-4)	45.31(5.72)
2	0.001165628	381	74	0.00316(7.8e-4)	76.92(3.42)
3	0.000614281	1214	105	0.00114(1.5e-4)	117.59(4.57)
4	0.000747053	835	234	0.00291(8.2e-4)	244.00(13.2)
5	0.001861556	428	63	0.00326(6.8e-4)	75.08(9.34)
6	0.000640149	1077	101	0.00186(3.5e-4)	102.49(4.08)
7	0.001561523	315	26	0.00440(8.5e-4)	32.78(5.55)
8	0.000151746	1040	124	0.00058(1.3e-4)	135.63(6.32)
9	0.000519559	933	71	0.00134(3.4e-4)	92.88(53.2)
10	0.000846462	858	55	0.00270(4.5e-4)	48.57(3.22)

Table 6.1: Results of 10 runs of  $(50 + 100)$ -ES on Case #1 (sinusoid). The standard deviation is given in brackets.



(a) Plot of the best solution (run #3) compared to the ideal solution for Case #2 (exponential). The MSE is 0.00363873.



(b) Plot of the fitness of the best solution (run #3) and the average fitness using  $(50 + 100)$ -ES for Case #2 (exponential).

Figure 6.3: The best solution of 10 runs on Case #2.

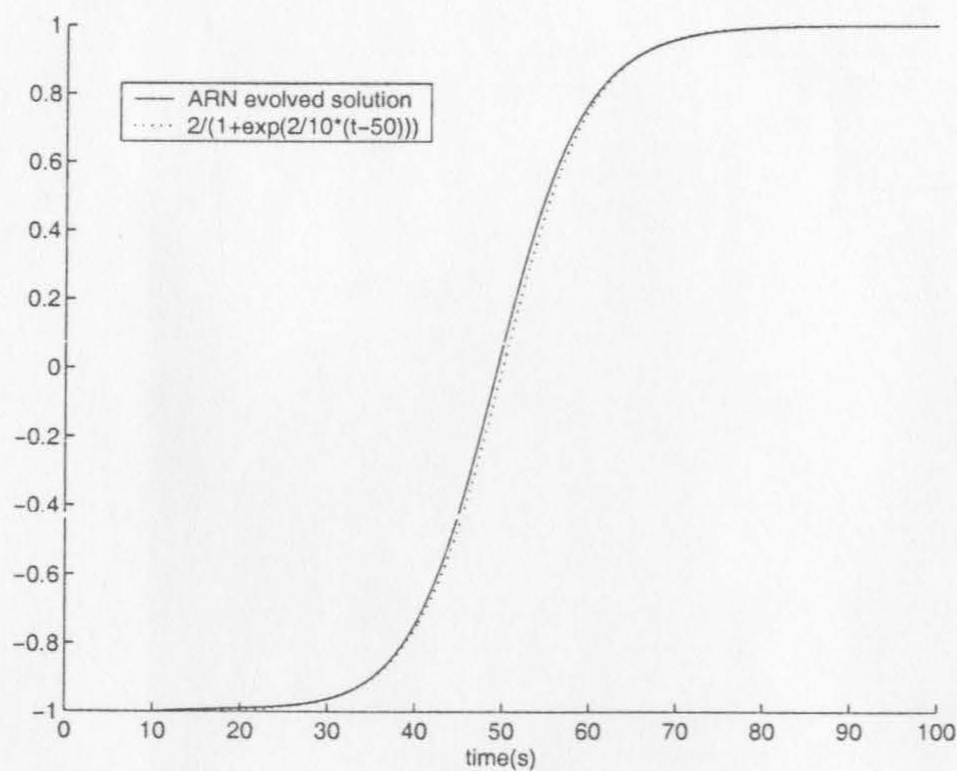
## 6.5 Analytical Considerations

We can see that a wide variety of networks with differing numbers of genes were found in the ARN framework to generate equivalent dynamics for the three fitness cases.

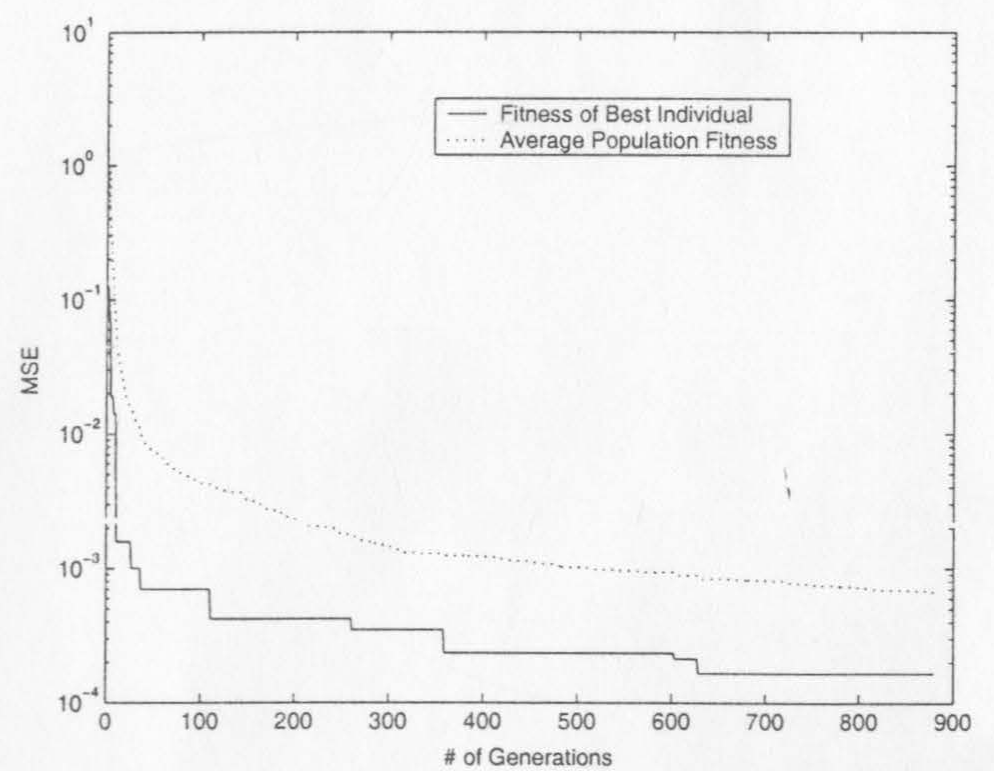


<i>Run #</i>	<i>Best MSE</i>	<i>#Gens.</i>	<i>#Genes</i>	<i>Avg. MSE(Pop.)</i>	<i>Avg. #Genes (Pop.)</i>
1	0.00411971	708	133	0.00447(1.3e-4)	142.83(5.88)
2	0.00478168	642	166	0.00554(2.5e-4)	185.95(13.5)
3	0.00363873	354	27	0.00641(5.5e-4)	52.22(7.00)
4	0.00441011	359	20	0.00660(6.1e-4)	31.95(7.38)
5	0.00381064	747	97	0.00505(3.0e-4)	106.81(5.71)
6	0.00402240	877	63	0.00464(1.8e-4)	58.83(4.17)
7	0.00426413	501	128	0.00574(3.5e-4)	116.14(8.75)
8	0.00537858	287	176	0.00661(4.6e-4)	164.40(11.1)
9	0.00511630	466	58	0.00688(5.6e-4)	54.26(3.73)
10	0.00588654	519	45	0.00643(1.7e-4)	45.65(3.10)

Table 6.2: Results of 10 runs of (50 + 100)-ES on Case #2 (exponential). The standard deviation is given in brackets.



(a) Plot of the best solution (run #4) compared to the ideal solution for Case #3 (sigmoid). The MSE is 0.0000173162.



(b) Plot of the fitness of the best solution (run #4) and the average fitness using (50 + 100)-ES for Case #3 (sigmoid).

Figure 6.4: The best solution of 10 runs on Case #3.



<i>Run #</i>	<i>Best MSE</i>	<i>#Gens.</i>	<i>#Genes</i>	<i>Avg. MSE(Pop.)</i>	<i>Avg. #Genes (Pop.)</i>
1	0.00101533	1235	154	0.00150(1.3e-4)	147.59(20.6)
2	0.00035992	557	36	0.00068(1.2e-4)	39.22(2.40)
3	0.00001843	758	100	0.00004(1.0e-5)	102.45(2.93)
4	0.00001732	721	96	0.00004(1.0e-5)	96.55(2.80)
5	0.00011328	617	97	0.00025(6.0e-5)	102.78(4.02)
6	0.00002073	825	104	0.00013(5.0e-5)	109.78(5.03)
7	0.00005429	465	108	0.00044(1.8e-4)	112.37(11.4)
8	0.00016598	879	177	0.00047(2.2e-4)	186.02(9.87)
9	0.00005034	575	195	0.00031(1.2e-4)	212.16(9.57)
10	0.00002219	987	39	0.00006(1.0e-5)	39.49(2.42)

Table 6.3: Results of 10 runs of (50 + 100)-ES on Case #3 (sigmoid). The standard deviation is given in brackets.

As can be seen from the results of the evolutionary runs, significantly large numbers of genes were used to obtain a solution. This is due to the fact that there was no penalty on the number of genes and no parsimony criterion during the evolution runs. To demonstrate this, the algorithm was run again with a penalty on the number of genes. The results of this are presented in Appendix I.

An interesting question to ask is, “What is the minimum number of genes required to generate equivalent dynamics for each fitness case?” In the case of the sinusoid, a simple oscillator can be written in the form,  $\ddot{y} + \omega^2 y = 0 \Leftrightarrow \ddot{y} = -\omega^2 y$  which describes the acceleration of an oscillating body. The acceleration is proportional but directed in the opposite direction to the displacement of the body. The equation also models a simple pendulum. When the pendulum crosses the vertical plane, an acceleration (i.e. gravity) pulls the pendulum in the opposite direction.

Defining,  $x_1 = \dot{y}$  and  $x_2 = \omega y$  and substituting this into the pendulum equation, we obtain  $\dot{x}_1 = \ddot{y} = -\omega^2 y = -\omega x_2$  and  $\dot{x}_2 = \omega \dot{y} = \omega x_1$ . Written as a matrix, this is:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} \mathbf{x}(t) \quad (6.1)$$



which leads to  $x_1 = -\sin(\omega t)$  and  $x_2 = -\cos(\omega t)$ .

We can take the vector  $x$  to be the concentrations of gene /protein pairs. So if Equation 6.1 was to be implemented in the ARN model how would it look? There would be two gene / protein pairs represented by nodes, "1" and "2". The first equation ( $\dot{x}_1 = \omega x_2$ ) can be implemented by node "2" having an inhibitory relationship with node "1". The second equation, likewise, can be implemented with an excitatory relationship between node "1" and node "2". In this way, the simple oscillator of Equation 6.1 can be implemented. For the ARN dynamic model to extract this oscillatory dynamic, it would simply have to have higher connectivity with one of the protein products of either node "1" or "2". Therefore, we can say that the minimum possible number of genes required to generate an oscillator in the ARN model is two. In fact, this is not surprising since we know that in order to generate an oscillator, we require a system which has entirely complex eigenvalues (with no real number component). Since the oscillator we desire is restricted to real numbers, we know that any complex eigenvalues must occur in complex conjugate pairs (indicating the minimum number of genes is 2). Therefore, this agrees with the previous analysis.

The requirements to generate a decaying exponential in the ARN model are decidedly simpler. In the dynamical equations the effects of excitation and inhibition on one gene are exponential in nature. Therefore, we simply need to have one gene in the system whose protein product binds with greater strength to the inhibitory rather than the excitatory site from which the dynamics are extracted. So, we simply need one gene in our system to create the dynamics of a decaying exponential.

The situation is somewhat more complicated in the case of the sigmoid type function. A means of deriving the minimum requirements for this function to a canonical form as was done for the previous two types of dynamics was not found. However, it can be reasoned that the minimum number of genes required must be greater than



one since a network with only one gene leads exclusively to exponential type dynamics. Appendix I demonstrates three different topologies of two-gene networks that generate sigmoid type dynamics. Therefore, it is reasonable to conclude that the minimum number of genes required in order to generate a sigmoid is two.

## 6.6 Conclusion

It has been demonstrated that the dynamics of a differential equation based ARN model created through duplication and divergence can be evolved toward simple functions. This suggests that such an approach may also be appropriate for generating arbitrary functions suitable for use in applications such as model optimization.

Due to the way in which the genes are detected on the genome, there are plentiful opportunities for individuals in the population to acquire neutral mutations which are beneficial in the context of evolution (Yu and Miller, 2001). Since there exist extensive non-coding regions of the genome, neutral mutations are free to be collected with new genes appearing suddenly when a new promoter pattern has been created through mutation. As well, each of the networks generated for each fitness case contains a different topology (number of genes). Therefore, due to the quality of solutions, it may be inferred that there are many different networks which can give good approximations to each of the fitness cases.

An open question within this framework is how the number of genes affects the ability to generate functions of a given type. However, from the results of this chapter, it is evident that it is quite easy to evolve the ARN model toward simple time series. In addition, it was seen that each evolved solution for any of the fitness cases differed largely from run to run. This would seem to indicate that there exist an extensive number of different topologies which can generate equivalent dynamics.



## Chapter 7

### Conclusion

This thesis has presented some of the methods for studying models of regulatory networks using mathematical and computational formalisms. In particular, an artificial regulatory network model first proposed by Banzhaf (2003a) was studied from the perspective of static network topology and the evolution of dynamics addressing questions raised in both artificial evolutionary processes and network biology.

Specifically, the model was examined from the standpoint of the scale-free, small-world and network motif topological properties when created using a whole genome duplication and divergence process. The whole genome duplication and divergence process was chosen since it has been previously implicated as an important factor in the evolution of genomes and due to its simplicity. Networks generated from this processes can in fact be classified as being scale-free and small-world. This is interesting since many researchers have claimed that the presence of scale-free and small-world network topologies are hallmarks of self-organization. In addition, these networks were also found to have subgraph distributions similar to those found in the transcriptional regulatory networks of *Escherichia coli* and *Saccharomyces cerevisiae* unlike those of random networks.



Since for all of these networks no evolution or modifications were made to the networks, the topologies obtained are directly related to the method of construction. This might indicate that such topologies may be artifacts of the method of creation rather than explicitly formed by evolution. Therefore, it may be more constructive in investigating transcriptional regulatory network topology to study the methods of network creation that nature has used. Efforts in this direction are just beginning. Even if the processes which create these networks have little to do with their subsequent topology, this thesis describes the effect of a whole genome duplication and divergence procedure on three different topological measures of networks specifically in the model of Banzhaf (2003a).

The evolution of dynamics of this model has also been investigated. The genome sequences could easily be evolved such that the dynamics generated matched those of simple output functions such as the sinusoid, sigmoid and decaying exponential functions. Examining the networks that were generated by the different genomes shows that many different networks give good approximations to each of the fitness cases. This would seem to indicate that within the ARN framework that there are an extensive number of different topologies which can generate equivalent dynamics which may be progressively evolved.



# Bibliography

- T. Aittokallio, M. Kurki, N. T., A. West, R. Lahesmaa, and O. Nevalainen. Computational strategies for analyzing data in gene expression microarray experiments. *Journal of Bioinformatics and Computational Biology*, 1(3):541–586, 2003.
- T. Akutsu. Identification of genetic networks from a small number of gene expression patterns under the boolean network model. In *Pacific Symposium on Biocomputing*, pages 17–28. World Scientific Press, 1999.
- T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for inferring qualitative models of biological networks. In *Pacific Symposium on Biocomputing*. World Scientific press, 2000a.
- T. Akutsu, S. Miyano, and S. Kuhara. Algorithms for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, pages 8–14. ACM Press, 2000b. ISBN 1-58113-186-0.
- R. Albert. Boolean modeling of genetic regulatory networks. In *The Third Conference on Parallel Problem Solving from Nature(PPSN)*, volume 650 of *Lecture Notes in Physics*, pages 459–481. Springer-Verlag, 2004.
- R. Albert, H. Jeong, and A.-L. Barabási. The internet’s achilles’ heel: Error and attack tolerance of complex networks. *Nature*, 406:378, 2000.
- B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. D. Watson. *Molecular Biology of the Cell*. Garland Science, 2002.



- S. Ando and H. Iba. Inference of gene regulatory model by genetic algorithms. In *Proceedings of the 3rd International Symposium on Adaptive Systems*, pages 15–22, 2001.
- S. Ando and H. Iba. Identifying the gene regulatory network by real-coded, variable-length, and multiple-stage GA, 2000.
- A. Arkin, P. Shen, and J. Ross. A test case of correlation metric construction of a reaction pathway from measurements. *Science*, 277:1275–1279, 1997.
- P. Augustsson, K. Wolff, and P. Nordin. Creation of a learning, flying robot by means of evolution. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1279–1285. Morgan Kaufmann Publishers, 9-13 July 2002. ISBN 1-55860-878-8.
- M. Babu and S. A. Teichmann. Evolution of transcription factors and the gene regulatory network in *Escherichia coli*. *Nucleic Acids Research*, 31(4):1234–1244, 2003.
- M. Babu, N. Luscombe, L. Aravind, M. Gerstein, and S. A. Teichmann. Structure and evolution of transcriptional regulatory networks. *Current Opinion in Structural Biology*, 14:283–292, 2004.
- W. Banzhaf. On the dynamics of an artificial regulatory network. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, *Advances in Artificial Life – Proceedings of the 7th European Conference on Artificial Life (ECAL)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 217–227. Springer-Verlag, 2003a.
- W. Banzhaf. Artificial regulatory networks and genetic programming. In R. L. Riolo and B. Worzel, editors, *Genetic Programming Theory and Practice*, chapter 4, pages 43–62. Kluwer, 2003b.
- W. Banzhaf and P. Kuo. Network motifs in artificial and natural transcriptional regulatory networks. *Journal of Biological Physics and Chemistry*, 4(2):85–92, 2004.
- A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- A.-L. Barabási, H. Jeong, R. Ravasz, Z. Neda, T. Vicsek, and A. Schubert. Evolution of the social network of scientific collaborations. *Physica A*, 311:590–614, 2002.



- A. Ben-Hur and H. Siegelman. Computation in gene networks. *Chaos*, 14(1):145–151, March 2004.
- J. Berg and M. Lassig. Local graph alignment and motif search in biological networks. *Proceedings of the National Academy of Sciences*, 101(41):14689–14694, 2004.
- D. D. Bernardo, T. Gardner, and J. Collins. Robust identification of large genetic networks. In *Pacific Symposium on Biocomputing*. World Scientific press, 2004.
- H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.
- J. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*, pages 1872–1877. IEEE Press, 2002.
- J. Bongard and H. Lipson. Automating genetic network inference with minimal physical experimentation using coevolution. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O’Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *GECCO 2004: Proceedings of the Genetic and Evolutionary Computation Conference*, volume 3242 of *Lecture Notes in Computer Science*, pages 333–345. Springer-Verlag, 2004.
- J. C. Bongard and R. Pfeifer. Evolving complete agents using artificial ontogeny. In F. Hara and R. Pfeifer, editors, *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, pages 237–258. Springer-Verlag, 2003.
- J. Bower and H. Bolouri, editors. *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, Cambridge, MA, 2001.
- D. Bray. Molecular networks: The top-down view. *Science*, 301:1864–1867, 2003.
- J. Brennecke, D. Hipfner, A. Stark, R. Russell, and S. Cohen. Bantam encodes a developmentally regulated microRNA that controls cell proliferation and regulates the proapoptotic gene *hid* in *drosophila*. *Cell*, 113(1):25–36, 2003.
- M. Brown, W. Grundy, D. Lin, N. Christianini, C. Sugnet, M. A. Jr., and D. Haussler. Support vector machine classification of microarray gene expression data. *UCSC-CRL 99-09, Department of Computer Science, University California Santa Cruz, Santa Cruz, CA*, 1999.



- H. Cao, L. Kang, Y. Chen, and J. Yu. Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, 1(4):309–337, 2000. ISSN 1389-2576.
- J. Carrington and V. Ambros. Role of microRNAs in plant and animal development. *Science*, 301(5631):336–338, 2003.
- N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, 2000.
- F. Chung, L. Lu, G. Dewey, and D. Galas. Duplication models for biological networks. *Journal of Computational Biology*, 10(5):677–687, 2003.
- G. Conant and A. Wagner. Convergent evolution of gene circuits. *Nature Genetics*, 34:264–266, 2003.
- M. Costanzo, M. Crawford, J. Hirschman, J. Kranz, P. Olsen, L. Robertson, M. Skrzypek, B. Braun, K. Hopkins, P. Kondu, C. Lengieza, J. Lew-Smith, M. Tillberg, and J. Garrels. YPD<sup>TM</sup>, PombePD<sup>TM</sup> and WormPD<sup>TM</sup>: model organism volumes of the BioKnowledge<sup>TM</sup> library, an integrated resource for protein information. *Nucleic Acids Research*, 29(1):75–79, 2001.
- E. Davidson. *Genomic Regulatory Systems*. Academic Press, San Diego, CA, 2001.
- D. Day and M. Tuite. Post-transcriptional gene regulatory mechanisms in eukaryotes: an overview. *Journal of Endocrinology*, 157(3):361–371, 1998.
- H. de Jong, J.-L. Gouzé, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathematical Biology*, 66(2):301–340, March 2004.
- P. D’Haeseleer, S. Liang, and R. Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.
- P. Dittrich, A. Burgel, and W. Banzhaf. Learning to move a robot with random morphology. In P. Husbands and J.-A. Meyer, editors, *Proceedings of the First European Workshop on Evolutionary Robotics*, volume 1468 of *Lecture Notes in Computer Science*, pages 165–178. Springer-Verlag, 16-17 Apr. 1998. ISBN 3-540-64957-3.



- R. Dobrin, Q. Beg, A.-L. Barabási, and Z. Olvai. Aggregation of topological motifs in the E. coli transcriptional regulatory network. *BMC Bioinformatics*, 5(10), 2004.
- R. Duda, P. Hart, and D. Stork. *Pattern Classification, 2nd edition*. Wiley Interscience, 2002.
- B. Dujon, D. Sherman, G. Fischer, P. Durrens, S. Casaregola, I. Lafontaine, J. De Montigny, C. Marck, C. Neuveglise, E. Talla, N. Goffard, L. Frangeul, M. Aigle, V. Anthouard, A. Babour, V. Barbe, S. Barnay, S. Blanchin, J. Beckerich, E. Beyne, C. Bleykasten, A. Boisrame, J. Boyer, L. Cattolico, F. Confanioleri, A. De Daruvar, L. Despons, E. Fabre, C. Fairhead, H. Ferry-Dumazet, A. Groppi, F. Hantraye, C. Hennequin, N. Jauniaux, P. Joyet, R. Kachouri, A. Kerrest, R. Koszul, M. Lemaire, I. Lesur, L. Ma, H. Muller, J. Nicaud, M. Nikolski, S. Oztas, O. Ozier-Kalogeropoulos, S. Pellenz, S. Potier, G. Richard, M. Straub, A. Suleau, D. Swennen, F. Tekaia, M. Wesolowski-Louvel, E. Westhof, B. Wirth, M. Zeniou-Meyer, I. Zivanovic, M. Bolotin-Fukuhara, A. Thierry, C. Bouchier, B. Caudron, C. Scarpelli, C. Gaillardin, J. Weissenbach, P. Wincker, and J. Souciet. Genome evolution in yeasts. *Nature*, 430(6995):35–44, 2004.
- B. Dutilh. *Analysis of data from microarray experiments, the state of the art in gene network reconstruction*. PhD thesis, Utrecht University, Theoretical biology and Bioinformatics, 1999.
- R. Edwards. Analysis of continuous-time switching networks. *Physica D*, 146:165–199, 2000.
- R. Edwards. Chaos in neural and gene networks with hard switching. *Differential Equations and Dynamical Systems*, 9:187–220, 2001.
- R. Edwards and L. Glass. Combinatorial explosion in model gene networks. *Chaos*, 10(3):691–704, 2000.
- R. Edwards, H. Siegelman, K. Aziza, and L. Glass. Symbolic dynamics and computation in model gene networks. *Chaos*, 11(1):160–169, March 2001.
- P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In I. Harvey and P. Husbands, editors, *Proceedings of the 4th European Conference on Artificial Life (ECAL)*, pages 205–213. MIT Press, 1997.



- E. Elowitz, A. Levine, E. Siggia, and P. Swain. Stochastic gene expression in a single cell. *Science*, 297:1183–1186, 2002.
- M. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, 2000.
- M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.
- A. Fire, S. Xu, M. Montgomery, S. Kostas, S. Driver, and C. Mello. Potent and specific genetic interference by double-stranded RNA in *caenorhabditis elegans*. *Nature*, 391:806–811, 1998.
- P. François and V. Hakim. Design of genetic networks with specified functions by evolution *in silico*. *Proceedings of the National Academy of Sciences*, 101(2):580–585, 2004.
- R. Friedman and A. L. Hughes. Gene duplication and the structure of eukaryotic genomes. *Genome Research*, 11(3):373–381, 2001.
- L. Galli-Taliadoros, J. Sedgwick, S. Wood, and H. Körner. Gene knock-out technology: a methodological overview for the interested novice. *Journal of Immunological Methods*, 181(1):1–15, 1995.
- T. Gardner, D. di Bernardo, D. Lorenz, and J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, 2003.
- L. Glass and S. Kauffman. The logical analysis of continuous, nonlinear biochemical control networks. *Journal of Theoretical Biology*, 39:103–129, 1973.
- K. Goh, E. Oh, H. Jeong, B. Kahng, and D. Kim. Classification of scale-free networks. *Proceedings of the National Academy of Sciences*, 99(20):12583–8, 2002.
- D. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems*, 3(5):493–530, 1989.
- Z. Gu, A. Cavalcanti, F.-C. Chen, P. Bouman, and W.-H. Li. Extent of Gene Duplication in the Genomes of *Drosophila*, Nematode, and Yeast. *Molecular Biology and Evolution*, 19(3):256–262, 2002.



- N. Guelzim, S. Bottani, P. Bourguin, and F. Képès. Topological and causal structure of the yeast transcriptional regulatory network. *Nature Genetics*, 31:60–63, 2002.
- I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2000.
- M. Hahn, G. Conant, and A. Wagner. Molecular evolution in large genetic networks: Does connectivity equal constraint? *Journal of Molecular Evolution*, 58:203–211, 2004.
- J. Hallinan and J. Wiles. Evolving genetic regulatory networks using an artificial genome. In Y.-P. P. Chen, editor, *Second Asia-Pacific Bioinformatics Conference (APBC2004)*, volume 29 of *CRPIT*, pages 291–296, Dunedin, New Zealand, 2004. ACS.
- S. Hammond, A. Caudy, and G. Hannon. Post-transcriptional gene silencing by double-stranded RNA. *Nature Reviews Genetics*, 2:110–119, 2001.
- D. Hand, H. Mannila, and P. Smyth. *Principals of Data Mining*. Cambridge, Massachusetts: MIT Press, 2001.
- I. Harvey and T. Bossamaier. Time out of joint: attractors in asynchronous random boolean networks. In P. Husbands and I. Harvey, editors, *Proceedings of the 4th European Conference on Artificial Life (ECAL)*, pages 67–75. MIT Press, 1997.
- J. Hasty. Design then mutate. *Proceedings of the National Academy of Sciences*, 99(26):16516–16518, 2002.
- J. Hasty, J. Pradines, M. Dolnik, and J. Collins. Noise-based switches and amplifiers for gene expression. *Proceedings of the National Acadamey of Sciences*, 97(4):2075–2080, 2000.
- L. Hood and D. Galas. The digital code of DNA. *Nature*, 421(6921):444–448, 2003.
- P. E. Hotz. Genome-physics as a new concept to reduce the number of genetic parameters in artificial evolution. In *Proceedings of the IEEE 2003 Congress on Evolutionary Computation*, pages 191–198. IEEE Press, 2003.
- A. L. Hughes. Gene duplication and the origin of novel proteins. *Proceedings of the National Academy of Sciences*, 102(25):8791–8792, 2005.



- A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999.
- A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000.
- T. Ideker, V. Thorsson, and R. Karp. Discovery of regulatory interactions through perturbation: inference and experimental design. In *Pacific Symposium on Biocomputing*. World Scientific press, 2000.
- F. Jacob, D. Perrin, C. Sanchez, and J. Monod. The operon: a group of genes whose expression is coordinated by an operator. *Comptes Rendus*, 250:1727, 1960.
- H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.
- H. Jeong, S. Mason, A.-L. Barabási, and Z. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.
- H. Kargupta. The gene expression messy genetic algorithm. In *Proceedings of the IEEE 1996 Congress on Evolutionary Computation*, pages 814–819. IEEE Press, 1996.
- H. Kargupta and S. Ghosh. Toward machine learning through genetic code-like transformations. *Genetic Programming and Evolvable Machines*, 3(3):231–258, 2002.
- N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics*, 20(11):1746–1758, 2004a.
- N. Kashtan, S. Itzkovitz, R. Milo, and U. Alon. Topological generalizations of network motifs. *Physical Review E*, 70:031909, 2004b.
- J. Kastner, J. Solomon, and S. Fraser. Modeling a *hox* gene network *in silico* using a stochastic simulation algorithm. *Developmental Biology*, 246:122–131, 2002.
- S. Kauffman. A proposal for using the ensemble approach to understand genetic regulatory networks. *Journal of Theoretical Biology*, 230:581–590, 2004.



- S. Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224: 177–178, 1969a.
- S. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22:437–467, 1969b.
- S. Kauffman. The large scale structure and dynamics of genetic control circuits: an ensemble approach. *Journal of Theoretical Biology*, 44:167–190, 1974.
- J. Keasling, H. Kuo, and G. Vahanian. A monte carlo simulation of the Escherichia coli cell cycle. *Journal of Theoretical Biology*, 176:411–430, 1995.
- M. Kellis, B. Birren, and E. Lander. Proof and evolutionary analysis of ancient genome duplication in the yeast *Saccharomyces cerevisiae*. *Nature*, 428:617 – 624, 2004.
- T. Kepler and T. Elston. Stochasticity in transcriptional regulation: Origins, consequences, and mathematical representations. *Biophysical Journal*, 81:3116–3136, 2001.
- P. M. Kim and B. Tidor. Limitations of Quantitative Gene Regulation Models: A Case Study. *Genome Research*, 13(11):2391–2395, 2003.
- H. Kitano, editor. *Foundations of Systems Biology*. MIT Press, Cambridge, MA, 2001.
- I. Kohane, A. Kho, and A. Butte. *Microarrays for an Integrative Genomics*. MIT Press, 2002.
- P. Kuo and W. Banzhaf. Scale-free and small world network topologies in an artificial regulatory network model. *Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE)*, pages 404–409, 2004.
- P. Kuo, A. Leier, and W. Banzhaf. Evolving dynamics in an artificial regulatory network model. In X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo-Guervós, J. Bullinaria, J. Rowe, P. Tino, A. Kabán, and H.-P. Schwefel, editors, *The Eighth Conference on Parallel Problem Solving from Nature (PPSN)*, volume 3242 of *Lecture Notes in Computer Science*, pages 571–580. Springer-Verlag, 2004.
- S.-I. Lee and S. Batzoglou. Application of independent component analysis to microarrays. *Genome Biology*, 4:R76, 2003.



- A. Leier, P. Kuo, and W. Banzhaf. Analysis of preferential network motif generation in an artificial regulatory network model created by duplication and divergence. *Advances in Complex Systems*, 2005, in preparation.
- S. Liang, S. Fuhrman, and R. Somogyi. REVEAL: A general reverse engineering algorithm for inference of genetic network architecture. In *Pacific Symposium on Biocomputing*, pages 18–29, 1998.
- W. Liebermeister. Linear modes of gene expression determined by independent component analysis. *Bioinformatics*, 18(1):51–60, 2002.
- N. Luscombe, M. Babu, H. Yu, M. Snyder, S. Teichmann, and M. Gerstein. Genomic analysis of regulatory network dynamics reveals large topological changes. *Nature*, 431:308–312, 2004.
- S. Mangan and U. Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.
- J. Mason, P. Linsay, J. Collins, and L. Glass. Evolving complex dynamics in electronic models of genetic networks. *Chaos*, 14(3):707–715, September 2004.
- J. Maynard Smith. *Evolutionary Genetics*. Oxford University Press, 1998.
- H. McAdams and A. Arkin. Stochastic mechanisms in gene expression. *Proceedings of the National Academy of Sciences*, 94:814–819, 1997.
- H. McAdams and A. Arkin. Simulation of prokaryotic genetic circuits. *Annual Review of Biophysics and Biomolecular Structure*, 27:199–224, 1998.
- T. Mestl, R. Bagley, and L. Glass. Common chaos in arbitrarily complex feedback networks. *Physical Review Letters*, 79(4):653–656, 1997.
- R. Milo, S. Shen-Or, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.
- R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Nature*, 303:1538–1542, 2004.



- J. Nadeau and D. Sankoff. Comparable rates of gene loss and functional divergence after genome duplications early in vertebrate evolution. *Genetics*, 147:1259–1266, 1997.
- H. Ochman, J. Lawrence, and E. Groisman. Lateral gene transfer and the nature of bacterial innovation. *Nature*, 405:299–304, 2000.
- S. Ohno. *Evolution by gene duplication*. Springer, Berlin, 1970.
- T. Perkins, M. Hallett, and L. Glass. Inferring models of gene expression dynamics. *Journal of Theoretical Biology*, 230:289–299, 2004.
- K. Polyak and G. J. Riggins. Gene Discovery Using the Serial Analysis of Gene Expression Technique: Implications for Cancer Research. *Journal of Clinical Oncology*, 19(11):2948–2958, 2001.
- M. Ptashne and A. Gann. *Genes and Signals*. Cold Spring Harbor Laboratory Press, 2001.
- C. Rangel, D. L. Wild, and F. Falciani. Modelling biological responses using gene expression profiling and linear dynamical systems. *Proceedings of the International Conference on System Biology*, pages 248–256, 2001.
- C. Rangel, J. Angus, Z. Ghahramani, M. Lioumi, E. Sotharan, A. Gaiba, D. L. Wild, and F. Falciani. Modeling T-cell activation using gene expression profiling and state-space models. *Bioinformatics*, 20(9):1361–1372, 2004a.
- C. Rangel, J. Angus, Z. Ghahramani, and D. Wild. Modeling genetic regulatory networks using gene expression profiling and state space models. In D. Husmeier, S. Roberts, and R. Dybowski, editors, *Probabilistic Modeling in Bioinformatics and Medical Informatics*, pages 269–293, 2004b.
- C. Rao, D. Wolf, and A. Arkin. Control, exploitation and tolerance of intracellular noise. *Nature*, 420:231–237, 2002.
- T. Reil. Dynamics of gene expression in an artificial genome: Implications for biological and artificial ontogeny. In D. Floreano, J.-D. Nicoud, and F. Mondada, editors, *Advances in Artificial Life – Proceedings of the 5th European Conference on Artificial Life (ECAL)*, volume 1674 of *Lecture Notes in Computer Science*, pages 457–466. Springer-Verlag, 1999.



- B. Reinhart, F. Slack, M. Basson, A. Pasquinelli, J. Bettinger, A. Rougvie, H. Horvitz, and G. Ruvkun. The 231-nucleotide let-7 RNA regulates developmental timing in *caenorhabditis elegans*. *Nature*, 403(6772):901–906, 2000.
- P. Romualdo, E. Smith, and R. Solé. Evolving protein interaction networks through gene duplication. *Journal of Theoretical Biology*, 222:199–210, 2003.
- A. Ryo, N. Kondoh, T. Wakatsuki, A. Hada, N. Yamamoto, and M. Yamamoto. A modified serial analysis of gene expression that generates longer sequence tags by nonpalindromic cohesive linker ligation. *Journal of Clinical Oncology*, 277(1):160–162, 2000.
- P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Res.*, 13(11):2498–2504, 2003.
- S. Shen-Or, R. Milo, S. Mangan, and U. Alon. Network motifs in the transcriptional regulation network of *Escherichia coli*. *Nature Genetics*, 31:64–68, 2002.
- T. Shi, D. Seligson, A. Beldegrun, A. Palotie, and S. Horvath. Tumor classification by tissue microarray profiling: random forest clustering applied to renal cell carcinoma. *Modern Pathology*, pages 1–11, 2004.
- M. L. Siegal and A. Bergman. Waddington’s canalization revisited: Developmental stability and evolution. *Proceedings of the National Academy of Sciences*, 99(16):10528–10532, 2002.
- A. Silvescu and V. Honavar. Temporal boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13:54–70, 2001.
- M. Skipper. RNA interference: Have our dreams been shattered? *Nature Reviews Genetics*, 4:671, 2003.
- R. Solé, R. Pastor-Satorras, E. Smith, and T. Kepler. A model of large-scale proteome evolution. *Advances in Complex Systems*, 5(1):43–54, 2002.
- P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the



- yeast *Saccharomyces cerevisiae* by microarray hybridization. *Molecular Biology of the Cell*, 9: 3273–3297, 1998.
- P. Swain. Personal communication. 2005.
- Z. Szallasi. Tutorial: Genetic network analysis – the perils and promises of massively parallel biology. *Intelligent Systems for Molecular Biology*, 2002.
- J. Taylor and J. Raes. Duplication and divergence: The evolution of new genes and old ideas. *Annual Review of Genetics*, 38(1):615–643, 2004.
- J. Tegnér, M. Yeung, J. Hasty, and J. Collins. Reverse engineering gene networks: Integrating genetic perturbations with dynamical modeling. *Proceedings of the National Academy of Sciences*, 100(10):5944–5949, 2003.
- S. A. Teichmann and M. Babu. Gene regulatory network growth by duplication. *Nature Genetics*, 36(5):492–496, 2004.
- C. Tennyson, H. Klamut, and R. Worton. The human dystrophin gene requires 16 hours to be transcribed and is cotranscriptionally spliced. *Nature Genetics*, 9(2):184–90, 1995.
- D. Thieffry, A. Huerta, E. Pérez-Rueda, and J. Collado-Vides. From specific gene regulation to global regulatory networks: a characterisation of *Escherichia coli* transcriptional network. *BioEssays*, 20:433–440, 1998.
- R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. i. structural conditions of multistationarity and other nontrivial behavior. *Chaos*, 11(1):170–179, March 2004a.
- R. Thomas and M. Kaufman. Multistationarity, the basis of cell differentiation and memory. ii. logical analysis of regulatory networks in terms of feedback circuits. *Chaos*, 11(1):180–195, March 2004b.
- S. Valverde, R. Ferrer Cancho, and R. Solé. Scale-free networks from optimal design. *Europhysics Letters*, 60:512–517, 2002.



- V. van Noort, B. Snel, and M. A. Huynen. The yeast coexpression network has a small-world, scale-free architecture and can be explained by a simple model. *EMBO Reports*, 5(3):280–284, 2004.
- A. Vazquez, R. Dobrin, D. Sergi, J.-P. Eckmann, Z. N. Oltvai, and A.-L. Barabasi. The topological relationship between the large-scale attributes and local interaction patterns of complex networks. *Proceedings of the National Academy of Sciences*, 101(52):17940–17945, 2004.
- E. Voit. *Computational Analysis of Biochemical Systems: A Practical Guide for Biochemists and Molecular Biologists*. Cambridge University Press, 2000.
- A. Wagner. How to reconstruct a large genetic network from  $n$  gene perturbation in  $n^2$  easy steps. *Bioinformatics*, 17:1183–1197, 2001.
- A. Wagner. Estimating coarse gene network structure from large-scale gene perturbation data. *Genome Research*, 12:309–315, 2002.
- A. Wagner. Evolution of gene networks by gene duplications: A mathematical model and its implications on genome organization. *Proceedings of the National Academy of Sciences*, 91:4387–4391, 1994.
- J. Watson, J. Wiles, and J. Hanan. Towards more relevant evolutionary models: Integrating an artificial genome with a developmental phenotype. In *Proceedings of the Australian Conference on Artificial Life (ACAL)*, pages 288–298, 2003.
- D. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton, NJ: Princeton University Press, 2003.
- D. Watts and S. Strogatz. Collective dynamics of small-world networks. *Nature*, 363:202–204, 1998.
- D. Weaver, C. Workman, and G. Stormo. Modeling regulatory networks with weight matrices. In *Pacific Symposium on Biocomputing*, 1999.
- K. Willadsen and J. Wiles. Dynamics of gene expression in an artificial genome. In *Proceedings of the IEEE 2003 Congress on Evolutionary Computation*, pages 199–206. IEEE Press, 2003.



- E. Winzeler, D. Shoemaker, A. Astromoff, H. Liang, K. Anderson, B. Andre, R. Bangham, R. Benito, J. Boeke, H. Bussey, A. Chu, C. Connelly, K. Davis, F. Dietrich, S. Dow, M. El Bakkoury, F. Foury, S. Friend, E. Gentalen, G. Giaever, J. Hegemann, T. Jones, M. Laub, H. Liao, N. Liebundguth, D. Lockhart, A. Lucau-Danila, M. Lussier, N. M'Rabet, P. Menard, M. Mittmann, C. Pai, C. Rebischung, J. Revuelta, L. Riles, C. Roberts, P. Ross-MacDonald, B. Scherens, M. Snyder, S. Sookhai-Mahadeo, R. Storms, S. Véronneau, M. Voet, G. Volckaert, T. Ward, R. Wysocki, G. Yen, K. Yu, K. Zimmermann, P. Philippsen, M. Johnston, and R. Davis. Functional characterization of the *s. cerevisiae* genome by gene deletion and parallel analysis. *Science*, 285(5429): 901–906, 1999.
- C. R. Woese. On the evolution of cells. *Proceedings of the National Academy of Sciences*, 99(13): 8742–8747, 2002.
- D. Wolf and A. Arkin. Motifs, modules and games in bacteria. *Current Opinion in Microbiology*, 6(2):125–134, 2003.
- K. Wolfe and D. Shields. Molecular evidence for an ancient duplication of the entire yeast genome. *Nature*, 387(6634):708–713, 1997.
- F. Wu, W. Zhang, and A. Kusalik. Modeling gene expression from microarray expression data with state-space equations. In *Pacific Symposium on Biocomputing*. World Scientific press, 2004.
- S. Wuchty. Scale-free behavior in protein domain networks. *Molecular Biology & Evolution*, 18(9): 1694–1702, 2001.
- S. Wuchty, Z. Oltvai, and A.-L. Barabási. Evolutionary conservation of motif constituents in the yeast protein interaction network. *Nature Genetics*, 35(2):176–179, 2003.
- A. Wuensche. Genomic regulation modeled as a network with basins of attraction. In *Pacific Symposium on Biocomputing*, pages 89–102, 1998.
- P. Xu, S. Vernooy, and B. Hay. The *drosophila* microRNA Mir-14 suppresses cell death and is required for normal metabolism. *Current Biology*, 13(9):790–795, 2003.
- M. Yamamoto, T. Wakatsuki, A. Hada, and A. Ryo. Use of serial analysis of gene expression (SAGE) technology. *Journal of Immunological Methods*, 250(1–2):45–66, 2001.



- E. Yeger-Lotem, S. Sattath, N. Kashtan, S. Itzkovitz, R. Milo, R. Y. Pinter, U. Alon, and H. Margalit. Network motifs in integrated cellular networks of transcription-regulation and protein-protein interaction. *Proceedings of the National Academy of Sciences*, 101(16):5934–5939, 2004.
- K. Yeung and W. Ruzzo. An empirical study on principal component analysis for clustering gene expression data. Technical report, UW-CSE, 2000.
- Y. Yokobayashi, R. Weiss, and F. Arnold. Directed evolution of a genetic circuit. *Proceedings of the National Academy of Sciences*, 99(26):16587–16591, 2002.
- Y. Yoshida and N. Adachi. A diploid genetic algorithm for preserving population diversity – pseudo-meiosis GA. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *The Third Conference on Parallel Problem Solving from Nature(PPSN)*, volume 866 of *Lecture Notes in Computer Science*, pages 36–45. Springer-Verlag, 1994. ISBN 3-540-58484-6.
- H. Yu, N. Luscombe, J. Qian, and M. Gerstein. Genomic analysis of gene expression relationships in transcriptional regulatory networks. *Trends in Genetics*, 19(8):422–427, 2003.
- H. Yu, D. Greenbaum, H. Lu, X. Zhu, and M. Gerstein. Genomic analysis of essentiality within protein networks. *Trends in Genetics*, 20(6):227–231, 2004.
- T. Yu and J. Miller. Neutrality and the evolvability of boolean function landscapes. In *Proceedings of the 4th European Conference on Genetic Programming (EuroGP)*, volume 2038 of *Lecture Notes in Computer Science*, pages 204–217. Springer-Verlag, 2001.
- J. Zhang. Evolution by gene duplication: an update. *Trends in Ecology and Evolution*, 18:292–298, 2004.
- D. Zill. *A First Course in Differential Equations with Modeling Applications*. Brooks Cole, 2000.



# Appendix A

## Measurement Technologies

High throughput technologies for obtaining mRNA expression levels such as genetic microarrays and the serial analysis of gene expression technique are briefly reviewed.

### A.1 Genetic Microarrays

In recent years, the number of scientific papers published using microarray technologies has grown tremendously. By enabling data acquisition on gene expression levels to be massively measured in parallel, genetic microarrays have helped to usher in the new age of so-called “systems biology”. As the pace of advancement in this field is staggering, any review of cutting-edge technology becomes quickly outdated. This section, presents some of the basic ideas and platforms of microarray technology.

All microarrays share the following: probes, and a substrate on which to deposit the probes. The probes are what is used to determine the expression level of a given mRNA. The details of the specific microarray platforms include the type of probe used (short oligo or long stretches of DNA), the substrate type (coated glass, polyacrilamide, etc...), how the probe is synthesized (in situ, or spotted directly), the



nature of the probe (ordinary nucleic acid or “locked nucleic acid”) and the labelled sample to be hybridized (cDNA or cRNA) (Szallasi, 2002). Two of the most common types of microarrays are cDNA arrays and oligonucleotide arrays. An overview of microarray technologies can be found in Kohane et al. (2002).

## A.2 Serial Analysis of Gene Expression (SAGE)

Serial analysis of gene expression (SAGE) is a method for quantifying gene expression within a given cell. The idea is to capture mRNA molecules within the cell, figure out what gene they were transcribed from, and then count the total number of mRNAs for each gene (allowing an estimate of how vigorously a given gene is being transcribed). The profiles of gene expression for different cell types are vastly different as are those of cancerous or infected cells. By studying these patterns of gene activity, researchers may be able to pinpoint the gene activity linked to particular diseases and conditions allowing for the development of targeted drug delivery.

Typically, mRNAs end with a long string of “A”s. In order to capture these mRNAs, microscopic magnetic beads are baited with strings of approximately 20 “T”s. Since “A”s and “T”s form a strong chemical bond, when the mRNAs are washed past a bath of these beads, the mRNAs become attracted to the beads. A magnet is then used to extract the beads and mRNAs out of the bath. These mRNAs are then copied back into DNA with the use of reverse-transcriptase.

These DNA fragments can then be identified using genetic sequencing. However, this means that one would have to sequence the DNA of every mRNA transcribed in the cell (an undertaking that could take decades using today’s technology). Luckily, a sequence of approximately 14-bases is required to identify a given gene. In order to speed the sequencing step further, each 14-base tag obtained from the reverse-



transcription of the mRNA is joined end-to-end to form a "concatemer". By subsequently amplifying the numbers of this concatemer (by duplication in bacteria), it can be later sequenced to obtain the number of times a given gene has been transcribed in the cell.

In summary, SAGE works by capturing RNA molecules, rewriting them as DNA, cutting a single 14-letter tag from the DNA, and joining all of these 14-letter tags together to form one long string of DNA. This long string is then sequenced allowing for the counting of the number of transcripts for each gene.

One of the primary advantages of SAGE is that it may be used to discover new genes. When studying an organism for which the complete genome is not available, if a tag is found which is not associated with a known gene, then it likely comes from a previously unknown gene. This has been exploited to find novel genes that could potentially play a significant role in tumourigenesis (Polyak and Riggins, 2001).

Although the use of tags of 14-bases is sufficient to uniquely identify genes, Ryo et al. (2000) have found that the use of 18-base tags leads to a more accurate DNA expression profile. There is also a problem with consistently obtaining the same number of bases from a given enzymatic cut. For instance, assuming tags of 14-bases, in a dibase of 28-bases, there is no way to be sure that this consists of two 14-base tags as opposed to one 16- and one 12-base tag. Yamamoto et al. (2001) have found that keeping the temperature constant greatly reduces such complications.

In addition, while it is true that most mRNAs end with "A"s, it is not true in all cases. Therefore, not all transcripts will be captured and will subsequently be left out of any analysis. Yamamoto et al. (2001) suggest the use of different combinations of anchoring and tagging enzyme be used to create two different profiles which can then be correlated and compiled representing the majority of genes expressed in the cell.



## Appendix B

# Determining Interactions Between Genes

Measuring mRNA levels gives us valuable data. However, such technologies do not tell us directly how one gene's products might interact with other genes. For this, various statistical, data mining, pattern recognition and machine learning methods are typically used. Since the vast majority of such analysis has been performed on microarray data, the information presented is more relevant to microarray analysis but is equally applicable to other mRNA measurement data. A good introduction to computational strategies for analyzing microarray experiments can also be found in Szallasi (2002) and Aittokallio et al. (2003).

### B.1 Correlation Analysis

Correlation analysis has long been a tool used in biological studies to generate hypotheses on possible causal relationships. In terms of the analysis of regulatory networks, the assumption behind correlation analysis is that genes which change their



expression levels in a correlated manner may possess some form of regulatory relationship. Obtaining a high correlation (or negative correlation) corresponds to four possible relationships – gene “A” regulates gene “B”, gene “B” regulates gene “A”, gene “A” and gene “B” are co-regulated by gene “C”, or chance. These relationships may also be effected through one or more intermediaries. However, it must be stressed that a high correlation (or negative correlation) is never proof of a causal relationship but should only be a means to propose hypotheses that must be tested by other methods. An example of a correlation metric used to reconstruct metabolic networks can be found in Arkin et al. (1997).

## B.2 Clustering Methods

In addition to correlation analysis methods, clustering and projection methods are also typically used to determine which protein products interact with other genes.

*K*-means and hierarchical clustering and their variants form one of the most common analysis methods for pattern recognition (Duda et al., 2002). As such, they have been used in past microarray studies (Spellman et al., 1998, Dutilh, 1999).

### B.2.1 *k*-means Algorithm

The *k*-means algorithm divides data samples (in this case genes) into *k* different clusters. Each cluster would theoretically represent those genes which are co-expressed and thus display some form of similarity. The algorithm works by randomly choosing *k* samples from the data set as the “centroid” of each cluster (each gene selected is assigned to a different cluster). Then each point in the remaining data set is assigned to a cluster based on its distance to the centroid of each cluster. After this assignment, the affected class centroid is updated. This process is often repeated several



times in order to gain some confidence in the results of the algorithm due to the inherent stochasticity present (in the initial class centroids, and sometimes in the order in which the remaining data points are assigned classes). A more complete treatment of the  $k$ -means algorithm can be found in (Duda et al., 2002).

### B.2.2 Hierarchical Clustering

In contrast to  $k$ -means clustering, hierarchical clustering methods are insensitive to initial conditions such as cluster number, prototype choice and sample ordering. Such methods function by developing a sequence of partitions where at each time step, partitions are combined to form sub-clusters. The criterion for joining together partitions are typically measures of minimum distance (nearest neighbour), maximum distance (furthest neighbour), or average distance. The merging of partitions can be displayed through the use of a dendrogram (a binary tree-like figure). A more complete treatment of hierarchical clustering can be found in Duda et al. (2002).

### B.2.3 Support Vector Machines

In addition to the traditional clustering algorithms, many machine-learning and classification approaches to clustering have been used with genetic data including random forests (a method which can be thought of as a boot-strapped version of hierarchical clustering) (Shi et al., 2004) and support vector machines (SVM) (Brown et al., 1999, Guyon et al., 2000). In the case of using classifiers on gene expression data, the classes would be analogous to clusters of co-expressed genes. SVMs has been shown to be successful in a variety of different applications such as text categorization, hand-written character recognition, image classification and bio-sequence analysis (Christianini and Shawe-Taylor, 2000).



SVMs emerged from work on the perceptron algorithm, another machine learning / discriminant algorithm (Hand et al., 2001). Earlier work on the perceptron learning rule focused on linearly separable classes. The best generalization performance for such systems was obtained when the discriminant hyperplane was as far as possible from the different classes of data points. The use of SVMs generalizes this concept even further allowing for non-linear decision surfaces which can perfectly separate the classes of data in the original measured feature space.

Another advantage to this method for developing classifiers is the lack of parameters required. Other methods such as maximum likelihood classifiers and maximum a posteriori methods all require the assumption of a probability distribution for the sampled data. Methods such as those related to neural networks require different choices of neuron types, learning rates (and methods), and network topologies. The SVM method does not require such parameters.

## **B.3 Projection Methods**

In contrast to clustering, projection methods attempt to find projections of the data which can prove to be more informative. A reduction in dimensionality may often be achieved with such methods. In this section, the methods of principal component analysis and independent component analysis are introduced.

### **B.3.1 Principal Component Analysis (PCA)**

PCA is a method by which a set of orthogonal feature vectors may be constructed in feature space where new features have zero correlation. In the case of genetic microarray data, the expression of each gene (across different samples) can be viewed as a separate feature vector in the original feature space. PCA then combines these



features in a linear fashion in order to generate new features which may be more informative.

The original data is then “projected” onto this new feature space where the eigenvalues represent the standard deviation of the data in the direction of the corresponding eigenvector (new feature direction). PCA is a simple data analysis tool. Yeung and Ruzzo (2000) have found that the features found via PCA may sometimes be uninformative in gene expression clustering. This is unsurprising since PCA restricts the feature space to being orthogonal (corresponding to the matrix eigenvectors). This restriction is removed in Independent Component Analysis (ICA).

### **B.3.2 Independent Component Analysis (ICA)**

ICA is an alternative method for analyzing and exploring large datasets. In a technical report by Yeung and Ruzzo (2000), it was found that PCA does not always find useful projections of the data. In certain cases, use of PCA on the data had a hugely detrimental effect on subsequent classification and machine learning schemes.

PCA enforces the constraints that the principal components must be orthogonal to each other, ICA allows non-orthogonal basis vectors. In PCA, one wishes to find rotations which lead to data which is uncorrelated when projected onto these new basis vectors. In ICA, one wishes to find rotations of the data which look as “non-Gaussian” as possible. Justifications, methods and a more detailed formulation of the ICA algorithm can be found in Hyvärinen (1999) and Hyvärinen and Oja (2000).

As was shown by Liebermeister (2002), the independent components of genetic microarray data can be directly related to distinct biological functions such as the phases of the cell cycle or the mating response and also has been used in microarray analysis (Lee and Batzoglou, 2003).



# Appendix C

## Subgraph Finding Algorithm

### C.1 Algorithm Implementation

In order to detect all  $n$ -node subgraphs, an algorithm similar to one devised by Milo et al. (2002) has been implemented. The algorithm of Milo et al. (2002) scans all rows of the adjacency matrix,  $M$ , searching for non-zero elements  $(i, j)$  which represent a connection from node  $i$  to node  $j$ . The algorithm then recursively traverses the neighbouring vertices connecting vertex  $i$  and  $j$  until a specific  $n$ -node subgraph is detected. The search traverses the graph disregarding edge direction (i.e. the algorithm may move from node “A” to node “B” even if the directed edge is from “B” to “A”). The constituent vertices and edges of a subgraph are then compared to previously found subgraphs in order to ensure that none have been over-counted. This form of search is analogous to depth first search (DFS) except the search process is terminated when a  $n$ -node subgraph is obtained. Three different data structure implementations were considered for the storage of the network topologies: an adjacency matrix, an edge-list and a node / edge list.

The adjacency matrix is one of the simplest schemes for storing network topology.



	<i>Storage Complexity</i>	<i>Search Complexity</i>
Adjacency Matrix	$n^2 = \frac{1}{4}e^2$	$2n$
Edge-List	$e$	$e$
Node / Edge List	$n + 2e = \frac{5}{2}e$	$c$

Table C.1: A comparison of the different data structure implementations considered.  $n$  is the number of nodes,  $e$  is the number of edges and  $c$  is a constant.

The matrix is typically composed of “0”s or “1”s in each of the  $(i, j)$  locations of the adjacency matrix. Every “1” in the adjacency matrix represents a connection from node  $i$  to node  $j$ . This is the approach taken by Milo et al. (2002). However, the storage requirements for this scheme are quadratic in the number of nodes ( $O(n^2)$ ). In addition, in order to perform the recursion required in the implementation of the search algorithm, the neighbours of each node must be determined. This entails examining each element in one row and one column of the adjacency matrix leading to the examination of  $2n$  entries anytime a neighbour of a node is to be determined.

An edge-list is another common method for storing a network topology. A typical implementation involves storing a pair of values which indicate a relationship between two nodes. For example, the pair  $(4, 3)$  would indicate that there exists an edge in the graph from node 4 to node 3. This leads to a storage requirement of  $O(e)$  where  $e$  represents the number of edges. However, a search for any neighbour of a given node requires a search over the entire list of edges. Therefore, the search complexity of using an edge list is also  $O(e)$ .

An alternative to the previous two implementations is a node / edge list. Each element of the data structure stores the node label and the labels of all incoming and outgoing edges. For instance, the element  $\{3, (2, 5, 6, 1), (7, 4)\}$  shows that node 3 has incoming edges from nodes 2, 5, 6 and 1 and edges which leave node 3 and terminate at nodes 7 and 4. This leads to a storage requirement of  $O(n + 2e)$ . However, the search complexity associated with the data structure is constant ( $O(1)$ ).



The storage and search complexity of each of the three data structures considered is presented in Table C.1. For the search complexity listed in Table C.1, the actual complexity of using a DFS algorithm is excluded since an equivalent DFS is implemented for each of the data structures considered. Therefore, the search complexity only gives the complexity directly related to the use of the specific data structure implementation with DFS. Since the types of graphs that were analyzed in this thesis typically have twice as many edges as nodes, Table C.1 has been rewritten with  $e = 2n$  to more easily facilitate comparison between the different implementations.

Therefore, we can see that the node / edge list data structure gives the best search performance with only a modest increase in storage requirements over the edge-list data structure. For this reason, the node / edge list data structure was used in implementing the subgraph enumeration algorithm.

It must be noted that the total number of subgraphs of a given type is counted and possible isomorphisms are considered the same subgraph type. The mappings of isomorphic graphs to a single canonical form is first performed in a preprocessing step. This step is accomplished by a brute force search through the  $2^{subgraphsize^2}$  possible adjacency matrices for isomorphisms. Of course, only connected graphs are considered by the algorithm where isomorphisms on an adjacency graph are found through the equivalence of row / column permutations. Isomorphisms are stored in a hash table for quick access during the actual search process. From this process, it was found that there were 86 non-isomorphic subgraphs of size three, 2818 types for networks of size four, 13930 types for networks of size five and 43700 types for networks of size three with inhibitory and excitatory connections.

The pseudocode for the particular implementation of the algorithm is given in Section C.2. Appendix E lists all three-node connection patterns in directed graphs, including auto-regulatory connections, up to isomorphism. Appendix G lists all four-



node connection patterns in directed graphs, including auto-regulatory connections, up to isomorphism. Particular subgraphs are referred to by their motif IDs (given in Appendices E and G).

## C.2 Algorithm Pseudocode

---

**Algorithm 2:** SearchIsomorphism: Find the set of all isomorphisms for all subgraphs of size  $n$ .

---

**input** : Size  $n$  of subgraphs

**output:** A mathematical set of all isomorphisms of size  $n$

$limit \leftarrow 2^{n \times n};$

$Set = \{canonical, isomorphism\};$

**for**  $i \leftarrow 1$  **to**  $limit$  **do**

$matrix \leftarrow \text{ConvertInttoMatrix}(i)$  ;

**if**  $\text{isMatrixConnected}(matrix)$  **then**

$reduced \leftarrow \text{ReduceMatrixtoSmallestInt}(matrix);$

$\text{AddToSet}(reduced, i);$

**end**

**end**

**return**  $Set$

---



---

**Algorithm 3:** ReduceMatrixtoSmallestInt: Reduction of a matrix into a canonical form integer.

---

**input** : Matrix -  $M$ , subgraph size -  $n$

**output:** canonical form integer -  $smallest$

$rowindex \leftarrow colindex \leftarrow \{1, 2, \dots, n\};$

$smallest \leftarrow \text{ConvertMatrixtoInt}(M);$

$permutest \leftarrow n!;$

**for**  $i \leftarrow 2$  **to**  $permutest$  **do**

$rowindex \leftarrow colindex \leftarrow \text{NextPermutation}(rowindex);$

$M2 \leftarrow \text{PermuteMatrix}(M, rowindex, colindex);$

$temp \leftarrow \text{ConvertMatrixtoInt}(M2);$

**if**  $temp < smallest$  **then**

$smallest \leftarrow temp;$

**end**

**end**

**return**  $smallest$

---

**Algorithm 4:** ObtainSubgraphCount: Obtain subgraph count for a graph  $G$ .

---

**input** : Adjacency matrix,  $G$  of a graph

**output:**

**for every directed edge**  $i$  **in**  $G$  **do**

$path \leftarrow \{\};$

$\text{AddEdgetoPath}(i, path);$

$\text{DepthFirstSearch}(i, path);$

**end**

---



---

**Algorithm 5:** DepthFirstSearch: Depth first search algorithm implementation

---

for subgraph counting.

---

**input** : next edge -  $i$ , repository of previously seen subgraphs -  $repos$

**output:**

AddEdgetoPath( $i, path$ );

if NumberofNodes( $path$ )  $\neq n$  then

$next \leftarrow$  NextEdgeinDFSpath;

    DepthFirstSearch( $next, path$ );

end

if SpecificSubGraphHasBeenPreviouslyCounted( $path, repos$ ) then

    return

end

AddPathtoRepository( $path, repos$ );

$matrix \leftarrow$  ConvertPathtoMatrix( $path$ );

$canonical \leftarrow$  ReduceMatrixtoSmallestInt( $matrix$ );

IncrementSubGraphCount( $canonical$ );

---



# Appendix D

## Parallelizing Motif Search

The runtime for search algorithms necessary for complete enumeration of the subgraphs of a network for a given size grows with network size (Kashtan et al., 2004a). Kashtan et al. (2004a) have devised a sampling algorithm which approximates the subgraph distributions of a given network. However, any sampling algorithm naturally introduces some errors (which decreases to zero when the sampling size is the same as the network size). An exact count of subgraph distributions is always preferable due to the higher degree of accuracy and confidence in the results. Thus, an investigation into possible parallelization of the algorithm is presented.

### D.1 Parallelization of the Subgraph Algorithm

Two methods for parallelizing the complete subgraph enumeration algorithm were considered. The first method was to use OpenMP to only slightly modify the serial version of the algorithm. Such an algorithm would be run under a shared memory paradigm on multiple processors. However, it should be noted that OpenMP does not support the parallelization of nested loops or recursions. Therefore, the implementa-



tion of the subgraph enumeration algorithm cannot be parallelized as given due to the recursive nature of the DFS algorithm. However, in order to investigate the effect of parallelization on the basic components of the system under a shared memory architecture, the search procedure was replaced with an inefficient search procedure which is easily parallelized. In this way, the effect of parallelization in general on parts of the algorithm can be qualitatively assessed. In addition, such an implementation is a trivial modification to the previously described recursive algorithm.

The DFS search procedure is replaced by a complete brute-force enumeration approach. The algorithm cycles through all permutations of nodes and checks whether a subgraph exists between these nodes (whether they are interconnected). If so, the subgraph type is determined and the count for that subgraph type is incremented. The pseudocode for such an approach is given in Algorithm 6.

---

**Algorithm 6:** TotalEnumerationSearch: Search the graph,  $G$ , using complete brute-force enumeration.

---

**input** : Adjacency matrix,  $G$  of a graph

**output:**

**for** every node  $i$  in  $G$  **do**

**for** every node  $j = i + 1$  in  $G$  **do**

**for** every node  $k = j + 1$  in  $G$  **do**

**if** IsConnected( $\{i, j, k\}$ ) **then**

$canonical \leftarrow \text{ReduceMatrixtoSmallestInt}(matrix);$

                IncrementSubGraphCount( $canonical$ );

**end**

**end**

**end**

**end**

---



Such an approach is needlessly computationally expensive requiring  $O\left(\begin{smallmatrix} n \\ k \end{smallmatrix}\right)$  computations where  $n$  is the number of nodes in the network and  $k$  is the subgraph size. In practice, this approach should always be outperformed by the recursive depth first search approach presented previously (an exception might be when networks are close to being fully connected). However, this method is easily parallelized since the outer-most of the triple-nested “for” loops can be divided amongst  $n$  processors. In this way, each processor can work on a different portion of the topology.

The second approach was to redesign the algorithm as a distributed memory program based on the message passing paradigm. Specifically, the MPI protocol was used with the algorithm being redesigned to run on the master / slave approach to load distribution. Each slave node possesses a copy of the complete graph topology. After each slave node has finished processing, the master node combines the results of each slave node’s processes (taking into account whether another slave node has previously enumerated a given subgraph). This involves updating a global histogram based on the results of each process. Although the maintenance of a separate copy of the graph topology in each process might be considered excessive in terms of memory requirements, an alternative is not easily implemented. Since it is not known apriori which nodes will be accessed by a given process, it is not possible to send only the required nodes to a given process. The only way to know which nodes are required is to actually perform the search leaving only the parallelization of the lookup for isomorphisms and the storage of the histogram (both of which take negligible time).

The algorithm is presented in Algorithms 7 and 8. It should be noted that the communication overhead has been minimized in the design of this algorithm. Since each processor owns its own copy of the topology, the only messages required to be sent from the master to the slave are the IDs for the edge from which to start



searching. Each slave node only passes back a confirmation that it has completed its operation on the given edge. When all edges have been searched, only then do the slave nodes send back the information that they have obtained from the search procedure. This information is then combined by the master node. In this way, much of the communication overhead inherent in the shared memory implementation of the algorithm may be avoided.

---

**Algorithm 7:** Master subroutine for the MPI version of the algorithm.

---

**input** : Edges of graph  $G$

**output:**

**for** *each processor*  $p$  **do**

    | `SendWork` ( $p, edge_i$ );

    |  $i \leftarrow i + 1$ ;

**end**

**for** *all remaining edges in*  $G$  **do**

    | `ReceiveResult` ( $p, junk$ );

    | `SendWork` ( $p, edge_i$ );

    |  $i \leftarrow i + 1$ ;

**end**

**for** *each processor*  $p$  **do**

    | `ReceiveResult` ( $p, junk$ );

**end**

`CombineResults` ();

---



---

**Algorithm 8:** Slave subroutine for the MPI version of the algorithm.

---

input : Edges of graph  $G$

output:

while do

    ReceiveWork ( $edge_i$ );

    if IsFinished() then

        WriteOutResults (*repository*);

        break;

    end

    DepthFirstSearch ( $edge_i$ );

end

---

No load balancing was used in the case of the shared memory algorithm, and no efforts at optimizing the block size of the work sent to each processor in the distributed memory algorithm were made.

## D.2 Comparison of Serial and Parallel Implementations of the Algorithm

In order to compare the performance of serial and parallel implementations of the subgraph enumeration algorithm, a test case was needed. A network was generated using the duplication and divergence process previously described that has been shown to have many of the topological properties found in biological networks. Specifically, the network generated from this process has 1700 nodes and 8918 edges. Such a network is somewhat small considering most networks under investigation such as the internet or genetic regulatory networks have at least tens of thousands of nodes (if not many more). However, a small enough test network was chosen in order to



be able to obtain a sufficient number of runs under both algorithm implementations and using different numbers of processors. It was also decided that the subgraph size would be limited to three since this is the size of the most common analysis in this field (Mangan and Alon, 2003, Milo et al., 2002, Shen-Or et al., 2002, Wolf and Arkin, 2003, Kashtan et al., 2004a, Kuo and Banzhaf, 2004, Milo et al., 2004, Vazquez et al., 2004) and would also reduce the running time of the algorithm.

### D.2.1 Shared Memory Algorithm

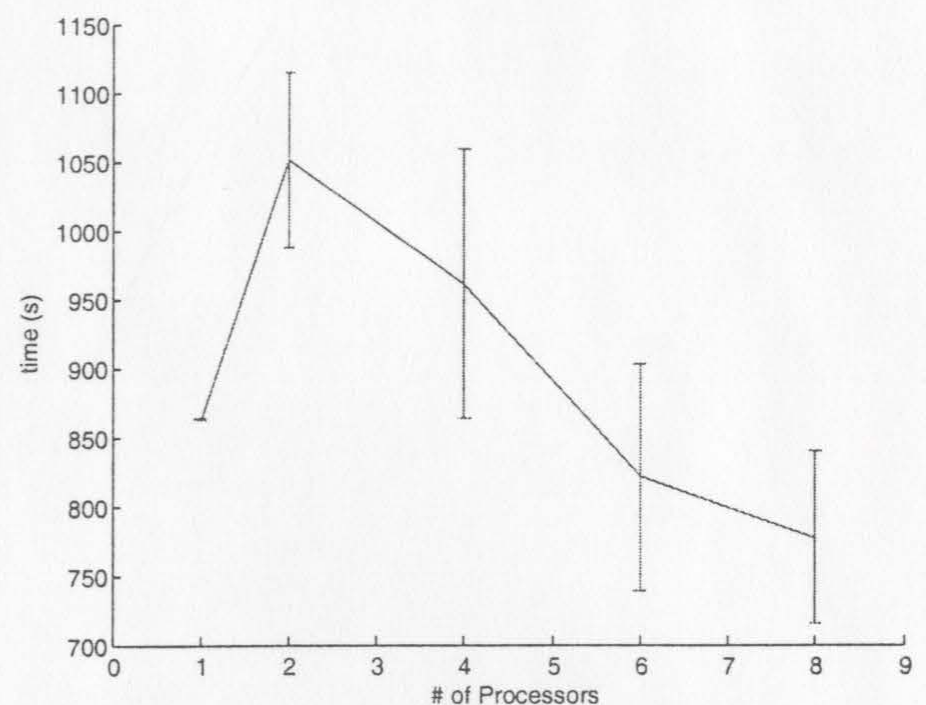
Figures D.1(a) and D.1(b) show the running times of the parallel implementations of the total enumeration algorithm using one, two, four, six and eight processors under the shared memory paradigm. The running time of the program for two processors is much more than that for one processor. In fact, the running time of the algorithm does not become consistently competitive with that of the serial version until the use of six processors. The efficiency attributed to parallelizing the algorithm can be obtained through a modification to Amdahl's Law, ( $Speedup = \frac{T(1)}{T(P)}$ ). Therefore, the efficiency is  $\frac{T(1)}{P * T(P)}$ , where  $T(1)$  is the running time of the algorithm on one processor and  $T(P)$  is the running time on  $P$  processors. The efficiency associated with the use of each number of processors is also given in Figure D.1(a). The efficiency of the algorithm implementation diminishes with an increase in the number of processors. This indicates that adding more processors to the implementation of the algorithm would have fewer and fewer benefits.

It should be noted that the shared memory program was written in C++ using the Intel C++ compiler (v8.0) with the following compiler options: -openmp, -O3 on an SGI Onyx 3400 (herzberg.physics.mun.ca). The number of processors used was restricted to eight since special permission is required from Memorial University of



Run #	# Processors				
	1	2	4	6	8
1	862.29	1093.50	900.04	854.64	704.17
2	862.69	989.96	1022.90	735.36	750.43
3	862.17	968.83	1019.00	689.50	752.01
4	862.73	1087.10	943.65	883.14	879.76
5	862.08	967.85	820.42	813.97	890.63
6	863.69	1087.90	1025.30	730.76	742.84
7	862.36	1088.50	949.99	953.13	694.97
8	862.13	967.25	1032.60	819.93	717.83
9	863.08	967.69	776.09	744.15	729.90
10	863.10	1088.40	912.75	699.01	885.35
11	863.16	1010.20	973.31	925.64	762.71
12	863.08	1099.20	1054.30	907.58	741.80
13	863.91	967.61	1105.70	910.90	802.82
14	861.61	1122.10	751.38	810.37	696.03
15	862.32	1100.60	910.39	898.80	826.99
16	862.75	1125.80	1040.50	864.69	746.39
17	862.54	1063.00	952.95	817.05	820.72
18	862.64	989.93	1034.00	828.68	790.99
19	862.32	1099.10	916.16	692.98	837.52
20	862.75	1128.10	1083.60	841.24	773.58
Mean	862.67	1050.63	961.25	821.08	777.37
Min	861.61	967.25	751.38	689.50	694.97
Max	863.91	1128.10	1105.70	953.13	890.63
Std. Dev.	0.555953	62.7457	97.85277	81.92894	62.1297
Efficiency	N/A	0.41	0.22	0.18	0.14

(a) Results of 20 runs of the complete brute-force enumeration algorithm using different numbers of processors under OpenMP on Herzberg.



(b) Plot of the average running time for  $n$  number of processors of the complete brute-force enumeration algorithm under OpenMP. The bars indicate the standard deviation of the running times on Herzberg.

Figure D.1: Performance of the shared memory algorithm.

Newfoundland Advanced Computation and Visualization Centre in order to use more processors for a given computation.

An obvious question is why we don't seem to see any benefit from parallelizing the shared memory algorithm until we use six processors? This is most likely due to the communications overhead required by the algorithm. Only one copy of the network topology is kept in shared memory. In addition, only one copy of the mapping of isomorphisms (which is relatively small), one copy of the list of previously searched motifs and one histogram are kept in shared memory. Every step of the algorithm accesses all of these elements which entails a high communication cost that was not present in the case of serial execution. Changing from the nested "for" loop structure of brute force enumeration used in the shared memory paradigm to the more efficient



recursive DFS routine would make little difference in relation to the communication overhead. Both routines require access to all of the aforementioned data structures at each step of the algorithm (the brute force method simply accesses these more often, but the DFS algorithm must still access these a significant number of times). Therefore, we can conclude that the current implementation is most likely not a good one for use on a shared memory architecture.

### D.2.2 Distributed Memory Algorithm

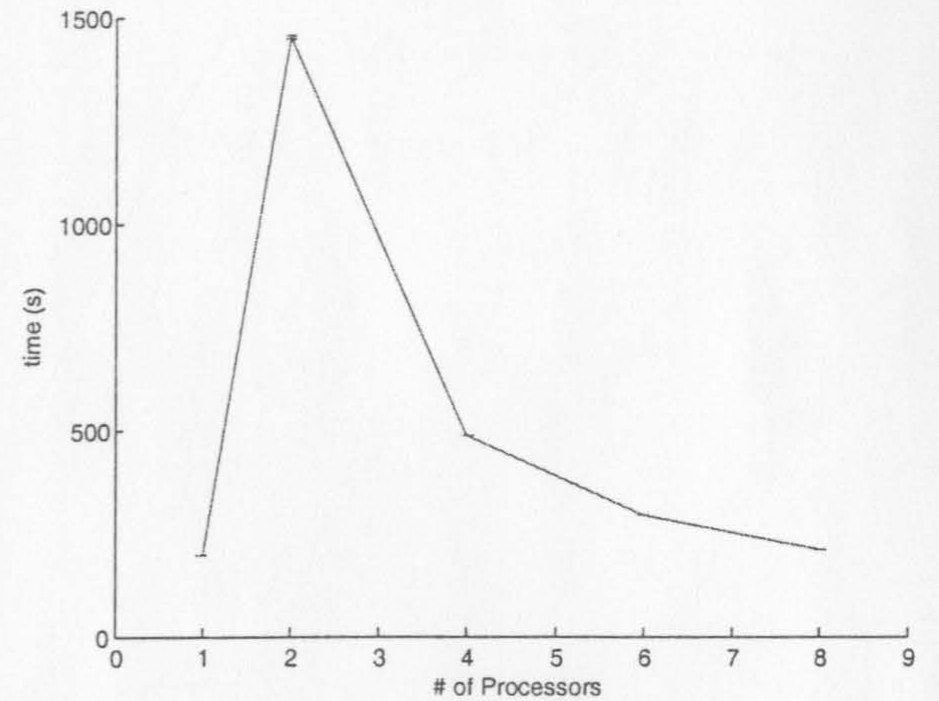
Figures D.2(a) and D.2(b) show the running times of the parallel implementations of the total enumeration algorithm using one, two, four, six and eight processors under the distributed memory paradigm. The running time of the program for two processors is again much more than that for one processor. In fact, the running time of the algorithm does not become competitive with that of the serial version until the use of eight processors. However, with an increasing number of processors, the efficiency of the computations increases. This would seem to indicate that this implementation is a better fit for the hardware architecture than that implemented for the shared memory architecture in the previous section. It can also be observed that this implementation of the algorithm is always faster than the previous shared memory implementation as can be seen from Figure D.1(a). This is to be expected since the distributed memory algorithm uses the more efficient DFS search while the shared memory algorithm uses a brute force enumeration approach.

It should be noted that the distributed memory program was written in C++ using the MPICH compiler (v1.2.6) with the following compiler options: `-lstdc++`, `-O3` on an SGI Onyx 3400 (herzberg.physics.mun.ca). The number of processors used was restricted to eight since special permission is required from Memorial University



Run #	# Processors				
	1	2	4	6	8
1	198.11	1447.70	484.38	290.79	208.69
2	195.99	1447.90	484.41	290.80	208.52
3	196.12	1447.60	484.40	290.88	208.41
4	196.67	1447.60	484.22	290.96	208.37
5	196.18	1447.90	484.08	290.75	208.46
6	197.22	1447.10	484.09	290.74	208.73
7	196.37	1448.10	484.19	290.71	208.47
8	197.72	1447.40	484.12	290.75	208.58
9	196.31	1464.50	484.33	290.77	208.59
10	196.35	1447.70	484.23	290.93	208.60
11	196.30	1447.50	484.23	290.86	208.66
12	197.00	1448.10	484.28	290.94	208.47
13	196.43	1447.50	484.22	290.72	208.66
14	195.86	1447.10	484.04	290.70	209.37
15	195.69	1447.20	484.29	290.78	208.79
16	196.16	1447.40	484.28	290.80	208.46
17	196.14	1447.40	484.66	290.88	208.70
18	196.71	1447.10	484.05	290.85	208.67
19	196.13	1447.80	484.23	290.69	208.69
20	196.12	1447.00	484.20	290.90	208.66
Mean	196.48	1448.38	484.25	290.81	208.63
Min	195.69	1447.00	484.04	290.69	208.37
Max	198.11	1464.50	484.66	290.96	209.37
Std. Dev.	0.612698	3.808522	0.146405	0.08448	0.21046
Efficiency	N/A	0.07	0.10	0.11	0.12

(a) Results of 20 runs of the recursive implementation of the algorithm using different numbers of processors under MPI on Herzberg.



(b) Plot of the average running time for  $n$  number of processors of the recursive implementation of the algorithm under MPI. The bars indicate the standard deviation of the running times on Herzberg.

Figure D.2: Performance of the distributed memory algorithm.

of Newfoundland Advanced Computation and Visualization Centre in order to use more processors for a given computation.

### D.3 Conclusions

From the results obtained from the two algorithm implementations we can conclude that using a parallel implementation of subgraph enumeration is not feasible. This may have more to do with the specific implementations of the algorithm than any inherent inability to effectively parallelize the search.

In the case of the shared memory algorithm, no attempt at load balancing was made. This might vastly increase the efficiency of the algorithm implementation. However, since the approach uses a brute force enumeration, it will always be outper-



formed by the DFS procedure. In fact, the performance of the serial version of the DFS algorithm outperforms the 8-processor version of the shared memory algorithm by almost an order of magnitude. Until recursive algorithms are supported under the OpenMP framework (or an efficient non-recursive search procedure is implemented), parallel implementations of subgraph enumeration on a shared memory architecture with OpenMP will remain impractical. The fact that the efficiency of the algorithm decreased when the number of processors was increased indicates that adding additional CPUs to the search process would have decreasing returns in running time.

In the case of the distributed memory algorithm, there are few ways to improve algorithm performance other than the adjustment of block size (amount of work sent during each request). This could significantly improve the running time of the algorithm. In fact, it is difficult to balance block size vs. load balancing. Sending larger block sizes might greatly increase performance possibly to the point that the parallel implementation outperforms the serial implementation. Since only a single work element is sent (which is often performed quickly), the amount of interprocess communication between the master and slave nodes is high with respect to the number of work requests (which is equivalent to the number of total edges in the graph). Specifically, there were 8918 work requests regardless of the number of processors. By increasing the amount of work sent at each communication step, there would be less time spent communicating between the master and slave thus potentially leading to drastic reductions in running time.

Neither of the two implementations presented seem to be especially promising as a replacement for the serial implementation of the subgraph enumeration algorithm. It may be the case that there does not exist a parallel algorithm which can effectively tackle this problem as is the case with some algorithms. However, before such a statement can be made more study of this problem and its parallelization is required.



## Appendix E

### Subgraphs of Size Three

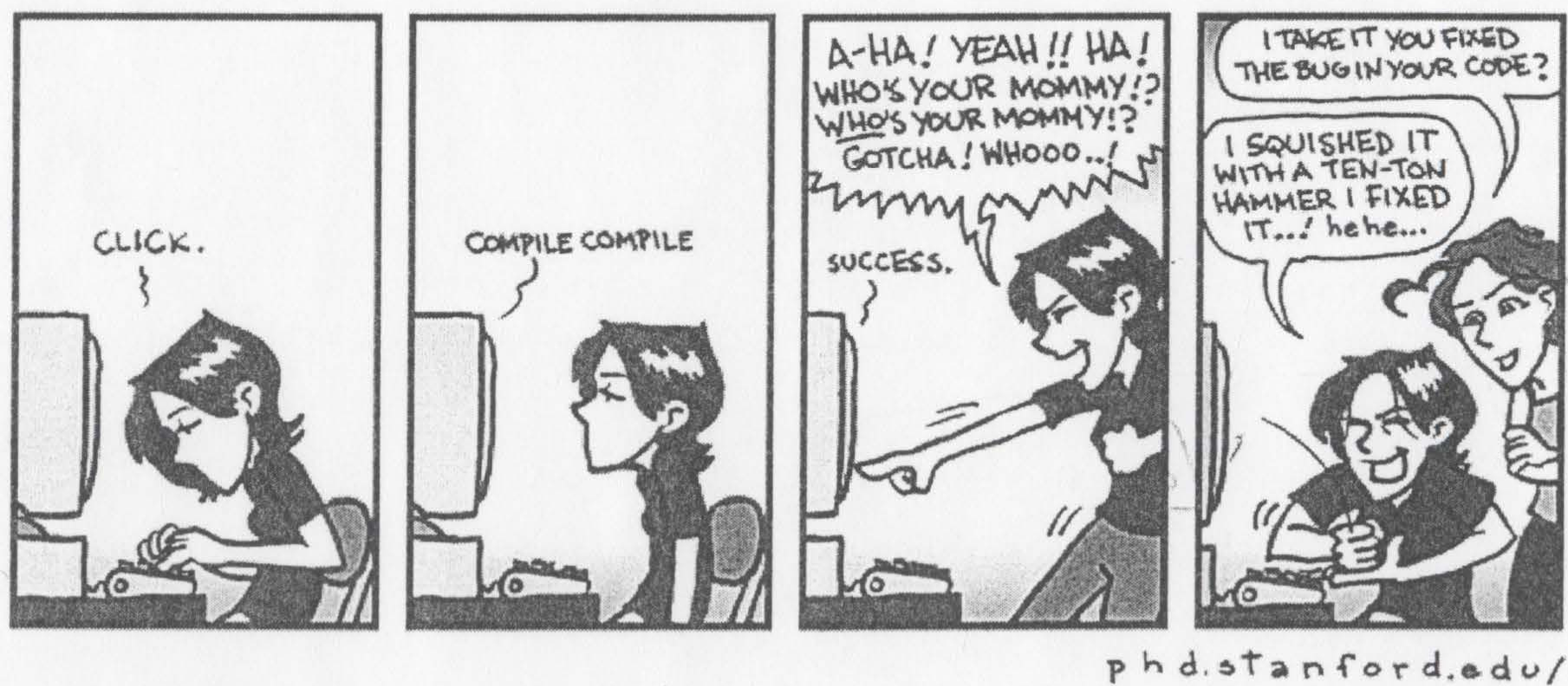


Figure E.1:



Motif ID	0	1	2	3	4	5	6	7	8	9
Motif ID	10	11	12	13	14	15	16	17	18	19
Motif ID	20	21	22	23	24	25	26	27	28	29
Motif ID	30	31	32	33	34	35	36	37	38	39
Motif ID	40	41	42	43	44	45	46	47	48	49
Motif ID	50	51	52	53	54	55	56	57	58	59
Motif ID	60	61	62	63	64	65	66	67	68	69
Motif ID	70	71	72	73	74	75	76	77	78	79
Motif ID	80	81	82	83	84	85				

Table E.1: Network motifs of size three and their ID.



## Appendix F

### Subgraph Counts for Size Three



Network IDs		Count in				Network IDs		Count in			
ID	ID*	D&D	Rand	<i>E. Coli</i>	<i>S. Cerv</i>	ID	ID*	D&D	Rand	<i>E. Coli</i>	<i>S. Cerv</i>
0	6	2424	76	35	751	45	A	1	0	0	0
1	A	4	0	0	1	46	110	0	0	0	0
2	12	490	271	40	246	47	A	0	0	0	0
3	A	11	0	26	24	48	A	0	0	3	0
4	14	6	0	0	0	49	A	0	0	0	0
5	A	0	0	0	0	50	A	0	0	0	0
6	A	12	0	124	138	51	A	0	0	0	1
7	A	0	0	8	0	52	A	0	0	0	0
8	A	0	0	1	0	53	A	0	0	1	0
9	A	0	0	2	0	54	A	0	0	0	0
10	A	0	0	0	0	55	A	0	0	0	0
11	A	0	0	0	0	56	A	0	0	0	0
12	36	27659	0	587	8800	57	A	0	0	0	0
13	A	8	0	76	104	58	A	0	0	0	0
14	38	15	0	2	44	59	A	0	0	54	4
15	A	0	0	1	1	60	A	0	0	12	0
16	A	20	0	11	22	61	A	0	0	0	0
17	46	0	0	0	1	62	A	0	0	0	0
18	A	0	0	0	0	63	A	0	0	0	0
19	A	0	0	2	1	64	A	10	0	0	0
20	A	0	0	1	0	65	A	0	0	0	0
21	A	0	0	0	0	66	A	0	0	0	0
22	A	5016	0	3353	2987	67	A	0	0	0	0
23	74	36	0	0	18	68	238	0	0	0	0
24	A	5	0	0	0	69	A	0	0	0	0
25	78	3	0	0	0	70	A	0	0	0	0
26	A	0	0	0	0	71	A	0	0	0	0
27	A	6	0	53	25	72	A	0	0	0	0
28	A	0	0	32	0	73	A	0	0	6	0
29	A	0	0	0	0	74	A	0	0	3	0
30	A	0	0	0	0	75	A	0	0	0	0
31	A	14	0	713	0	76	A	0	0	46	0
32	A	0	0	0	0	77	A	0	0	0	0
33	A	3	0	0	0	78	A	0	0	0	0
34	A	0	0	0	0	79	A	0	0	0	0
35	A	0	0	0	0	80	A	0	0	0	0
36	A	0	0	0	0	81	A	0	0	0	0
37	A	0	0	0	0	82	A	0	0	0	0
38	98	0	0	0	0	83	A	0	0	0	0
39	A	0	0	0	0	84	A	0	0	0	0
40	102	0	0	0	0	85	A	0	0	0	0
41	A	0	0	0	0						
42	A	6	0	14	3						
43	A	0	0	0	0						
44	108	0	0	0	0						

Table F.1: Subgraphs of size three and their distribution. D&D: Duplication and divergence genomes; Rand: Random genomes. ID\* are the subgraph designations given by Milo et al. (2002). IDs shown as A are subgraphs with self-regulatory connections which do not have a designation in Milo et al. (2002).



# Appendix G

## Subgraphs of Size Four

### DATA: BY THE NUMBERS



[www.phdcomics.com](http://www.phdcomics.com)

Figure G.1:



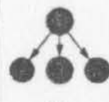



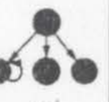











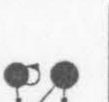


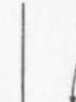
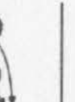






















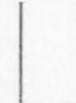
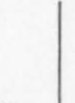











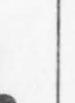

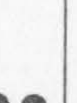






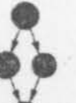

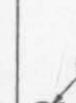
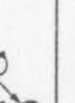



Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												

Table G.1: Subgraphs of size four and their ID. Only motifs which were present in at least one of the four cases are shown. All other motifs have been omitted.



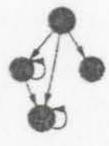
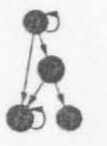
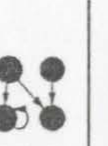




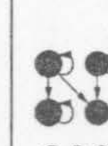
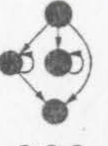








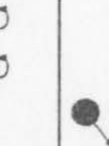


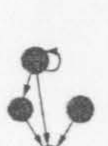



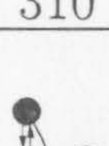


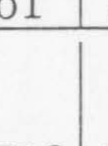
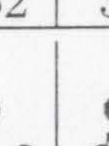
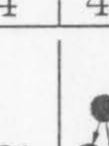

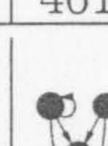
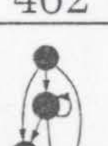
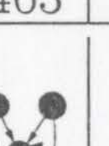


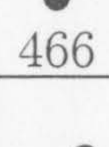
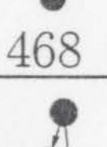
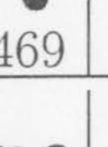
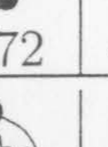
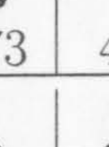
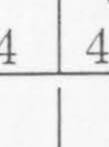
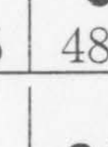
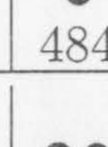
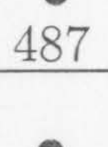
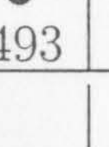
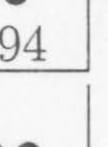



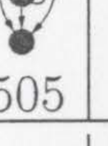
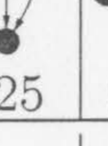
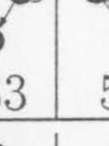
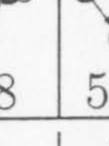
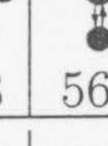



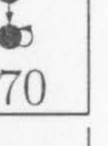





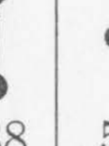
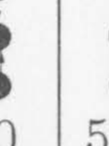










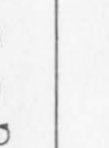
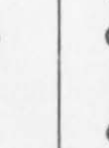



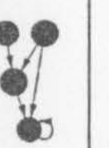














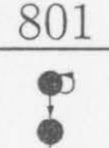
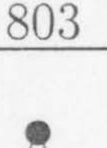
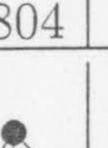
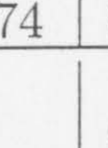
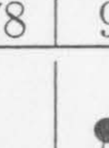
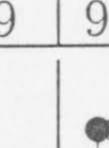
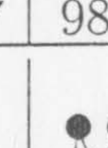
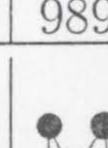
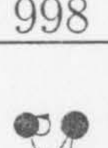

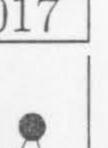

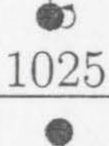
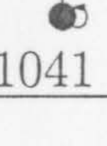
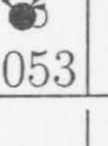
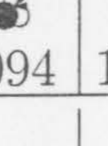
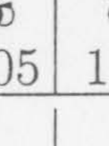
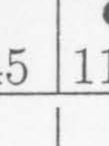
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												
Motif ID												

Table G.2: Subgraphs of size four and their ID. Only motifs which were present in at least one of the four cases are shown. All other motifs have been omitted.



## Appendix H

### Subgraphs Counts for Size Four



Network IDs	Count in				Network IDs	Count in			
	D&D	Rand	<i>E.Coli</i>	<i>S.Cerv</i>		D&D	Rand	<i>E.Coli</i>	<i>S.Cerv</i>
0	4137	43	4	843	114	0	0	0	1
2	56	125	10	116	120	0	0	12	0
3	0	1	0	5	123	0	3	18	43
4	1716	2	0	0	124	0	0	1	0
6	3	2	38	150	125	0	0	0	5
8	0	2	0	0	126	0	0	1	0
12	61	249	3	329	131	0	0	259	0
13	0	3	0	0	137	0	0	1	0
14	1531	247	510	16925	145	1	4	10	27
15	0	3	0	31	150	2	4	0	10
16	9	5	0	75	154	1	0	0	0
18	0	3	5	19	158	10	0	7	14
19	0	2	0	0	164	0	0	0	1
21	0	4	1	11	199	0	3	6	28
22	0	0	0	3	200	0	0	14	0
23	1	0	0	0	201	0	0	5	3
26	0	3	36	157	202	0	0	1	0
28	0	0	2	10	207	0	0	5	0
35	1337	1	8	1105	237	39	2	0	6
37	0	0	0	5	273	0	0	40	2
39	0	0	0	1	274	0	0	6	0
45	1451	123	118	1246	275	0	0	1	0
46	0	1	72	81	279	0	0	9	0
47	10	4	0	0	281	0	0	508	0
49	530	0	0	0	282	0	0	30	0
51	0	4	58	4	283	0	0	1	0
55	0	3	1	0	289	0	0	1	0
56	0	0	6	0	293	1	4	704	1261
63	10	245	0	92	294	0	0	16	0
64	0	3	8	0	295	0	0	0	2
65	0	4	0	0	296	0	0	1	0
67	0	4	0	0	298	0	0	1	0
69	1	0	0	0	301	0	0	43	14
71	0	5	0	11	302	0	0	3	0
77	1	0	0	0	303	0	0	7	0
79	0	4	0	0	306	0	0	1	0
88	0	0	1	0	309	6	0	125	737
95	0	4	7	0	310	0	0	5	0
96	1	4	0	0	342	0	4	4	0
98	1293	246	188	3859	343	0	0	11	0
99	0	3	167	528	361	0	0	1	0
100	0	5	0	51	362	0	0	1	0
102	1	4	0	0	364	0	0	1	0
106	291	3	3569	4618	459	301970	41	2052	88321
108	2	4	0	16	460	8	1	391	1085
112	1	4	1	195	461	157	4	25	729
113	0	0	39	83	462	2	0	8	23

Table H.1: Subgraphs of size four and their distribution. D&D: Duplication and divergence genomes; Rand: Random genomes. Only motifs which were present in at least one of the four cases are shown.



Network IDs	D/D	Count in Rand	<i>E.Coli</i>	<i>S.Cerv</i>	Network IDs	D/D	Count in Rand	<i>E.Coli</i>	<i>S.Cerv</i>
463	1	0	0	1	786	0	0	1950	118
465	1	0	46	346	787	2	0	96	3
466	0	0	0	9	788	0	0	11	0
468	0	0	0	1	801	167	0	659	0
469	0	0	0	1	803	75	0	0	0
472	0	0	17	6	804	0	0	0	1
473	0	0	9	0	974	0	0	18	0
474	0	0	3	2	978	0	0	15	0
475	0	0	2	0	979	0	0	9	0
483	4	0	0	120	987	0	0	2	0
484	0	0	1	1	988	0	0	202	0
487	0	0	0	1	989	0	0	81	0
493	5	0	16	33	998	0	0	281	0
494	0	0	0	17	1001	0	0	1	0
498	0	0	1	4	1017	0	0	1	0
499	0	0	0	15	1025	0	0	1	0
505	0	0	1	0	1041	0	0	15	1
525	0	0	0	1	1053	0	0	9	1
533	0	0	0	2	1094	0	0	2710	0
548	0	0	1	0	1105	0	0	124	0
563	130570	0	45585	59569	1145	0	0	61	0
564	521	2	0	121	1160	0	0	13	0
565	34	0	0	0	1521	44	0	26	3
566	11	0	0	0	1526	5	0	0	0
568	54	0	0	0	1531	0	0	9	0
570	16	2	191	129	1606	0	0	6	0
571	0	0	103	0	1612	0	0	0	1
576	161	0	19077	0	1618	0	0	5	0
578	20	0	0	0	1846	0	0	57	1
587	410	3	1606	150	1847	43	0	7	0
588	8	4	0	0	1855	354	0	0	0
590	24	2	0	32	1897	0	0	14	0
594	3	4	0	0	1898	0	0	4	0
602	1028	0	415	24	1957	0	0	208	0
606	27	0	0	0	1958	0	0	1	0
617	0	0	90	0	1968	0	0	99	0
622	0	0	0	16	2094	0	0	14	0
632	0	0	5	0	2339	0	0	1	0
647	3	0	0	0	2486	0	0	8	0
654	2	0	0	0	2579	1	0	0	0
658	20	0	0	0	2619	0	0	4	0
691	0	0	624	0	2623	0	0	30	0
692	0	4	6	0	2634	0	0	1	0
693	0	0	8	0	2643	0	0	18	0
695	0	0	7	0	2677	0	0	120	0
722	0	0	0	1					
750	0	1	0	0					

Table H.2: Subgraphs of size four and their distribution. D&D: Duplication and divergence genomes; Rand: Random genomes. Only motifs which were present in at least one of the four cases are shown.



# Appendix I

## Evolving Networks with a Restricted Number of Genes

The algorithm in Section 6.3 was reapplied with the addition of a penalty on the number of genes. Since penalty functions are typically arbitrary and problem dependent (since they directly affect the search space), a simple approach was taken. Instead of penalizing the number of genes in the system, networks with more than 10 genes were set to have a fitness of 4.0. In this way, the fitness landscape of each fitness case is not as directly impacted as would be the case if a penalty was implemented commensurate with the number of genes. This may cause various basins of attraction to become isolated on the fitness landscape.

Results of 10 runs on each fitness case are shown in Tables I.1, I.2, I.3. In addition, the evolution algorithm was terminated when the best fitness obtained was below  $5.0 \times 10^{-3}$  rather than after 250 generations of fitness stagnation. This was done in order to show that reasonable solutions to the problem may be obtained with this modified fitness function. Use of the previous fitness function can lead to algorithm termination before a good solution has been obtained.



<i>Run #</i>	<i>Best MSE</i>	<i>#Gens.</i>	<i>#Genes</i>	<i>Avg. MSE(Pop.)</i>	<i>Avg. #Genes (Pop.)</i>
1	0.00287157	89122	10	0.00734(1.1e-3)	9.73(0.54)
2	0.00444153	13643	8	0.00912(8.1e-4)	7.29(0.43)
3	0.00486211	401417	9	0.01027(2.3e-4)	9.18(0.18)
4	0.00470516	133229	10	0.00707(6.1e-4)	10.20(0.20)
5	0.00356387	21205	10	0.01493(4.7e-3)	10.20(0.20)
6	0.00493755	99553	10	0.00870(1.5e-3)	9.92(0.49)
7	0.00398828	11342	10	0.02751(1.3e-2)	10.00(0.49)
8	0.00472991	23091	10	0.00989(2.4e-3)	10.20(0.20)
9	0.00480238	395	9	0.30263(7.5e-2)	9.47(0.56)
10	0.00281274	1664	8	0.20032(7.5e-2)	9.59(0.89)

Table I.1: Results of 10 runs of (50 + 100)-ES on Case #1 (sinusoid) with a penalty function. The standard deviation is given in brackets.

<i>Run #</i>	<i>Best MSE</i>	<i>#Gens.</i>	<i>#Genes</i>	<i>Avg. MSE(Pop.)</i>	<i>Avg. #Genes (Pop.)</i>
1	0.00484099	639	8	0.00811(5.4e-4)	7.02(2.08)
2	0.00492588	2799	9	0.00714(6.2e-4)	9.02(0.98)
3	0.00418354	820	5	0.00659(5.0e-4)	6.32(1.69)
4	0.00478972	5336	9	0.00636(4.9e-4)	9.33(1.02)
5	0.00497284	1676	9	0.00759(4.2e-4)	9.31(0.71)
6	0.00490717	468	9	0.00810(6.9e-4)	8.82(1.01)
7	0.00430360	642	10	0.00785(6.5e-4)	8.51(1.49)
8	0.00472030	3529	10	0.00577(2.6e-4)	9.67(0.73)
9	0.00467765	10112	10	0.00601(2.6e-4)	10.18(0.25)
10	0.00413019	241	5	0.00798(9.1e-4)	7.00(1.66)

Table I.2: Results of 10 runs of (50 + 100)-ES on Case #2 (exponential) with a penalty function. The standard deviation is given in brackets.

<i>Run #</i>	<i>Best MSE</i>	<i>#Gens.</i>	<i>#Genes</i>	<i>Avg. MSE(Pop.)</i>	<i>Avg. #Genes (Pop.)</i>
1	0.00345716	35	6	0.05491(1.8e-2)	8.84(1.35)
2	0.00375144	61	9	0.04274(1.5e-2)	8.80(1.05)
3	0.00425317	8	6	0.13660(7.1e-2)	7.71(1.66)
4	0.00149893	15	8	0.10153(4.1e-2)	8.41(1.62)
5	0.00373932	21	10	0.07446(3.5e-2)	8.44(1.42)
6	0.00299901	208	8	0.01359(4.0e-3)	8.92(0.99)
7	0.00341115	32	7	0.03841(1.1e-2)	8.55(1.16)
8	0.00492678	109	10	0.01886(6.7e-3)	8.49(1.25)
9	0.00101274	4	6	0.39698(1.8e-1)	7.73(1.84)
10	0.00423338	19	9	0.07139(3.1e-2)	8.59(1.40)

Table I.3: Results of 10 runs of (50 + 100)-ES on Case #3 (sigmoid) with a penalty function. The standard deviation is given in brackets.



In order to show that the sigmoid dynamics can be generated with two genes, the algorithm was rerun such that networks with more than two genes had a fitness of 4.0. Figure I.1 shows examples of three different network topologies which can generate the sigmoid dynamics.

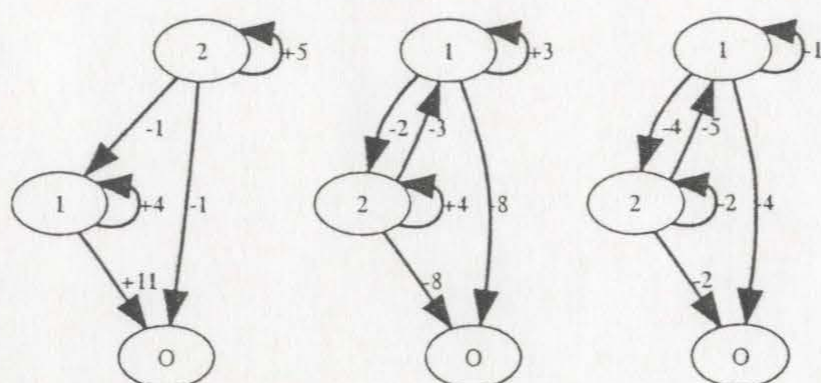


Figure I.1: Three two-gene networks that generate sigmoid dynamics. The “O” node denotes the additional site used to extract the network dynamics.







